Numerical Optimization with Applications

Suresh Chandra
Jayadeva
Aparna Mehra

Narosa

Numerical Optimization with Applications

Numerical Optimization with Applications

Suresh Chandra
Jayadeva
Aparna Mehra

Narosa Publishing House
New Delhi Chennai Mumbai Kolkata

Numerical Optimization with Applications

710 pgs. | 113 figs.

Suresh Chandra
Department of Mathematics

Jayadeva
Department of Electrical Engineering

Aparna Mehra
Department of Mathematics

Indian Institute of Technology Delhi
New Delhi, India

Second Reprint 2016

NAROSA PUBLISHING HOUSE PVT. LTD.

22 Delhi Medical Association Road, Daryaganj, New Delhi 110 002
35-36 Greams Road, Thousand Lights, Chennai 600 006
306 Shiv Centre, Sector 17, Vashi, Navi Mumbai 400 703
2F-2G Shivam Chambers, 53 Syed Amir Ali Avenue, Kolkata 700 019

www.narosa.com

Printed from the camera-ready copy provided by the Author.

ISBN 978-81-7319-854-0

Published by N.K. Mehra for Narosa Publishing House Pvt. Ltd.,
22 Delhi Medical Association Road, Daryaganj, New Delhi 110 002

Printed in India

The first author (S. Chandra) would like to dedicate this book to the loving memory of his son, Suyash, who had been the main motivating force for writing this book.

The second author (Jayadeva) would like to dedicate this book to his parents, Shri R. Sampat Kumaran and Smt. M. C. Ramananda, who have been a lifelong source of inspiration and support.

The third author (A. Mehra) dedicates this book to the beautiful memories of her mother Smt. Karuna Mehra, care of her father Shri Alok K. Mehra, and faith of her teachers.

Preface

This book has evolved from certain courses on optimization that we have been teaching at I.I.T. Delhi for the last so many years.

The main aim of this book is to provide a focussed and detailed study of various numerical optimization methods and their applications in science, engineering, finance and management. Apart from discussing standard optimization methods, the book includes some very recent topics like *semi-definite programming, second order cone programming and evolutionary methods for global optimization*. An attempt has also been made to present some modern and nonconventional applications of numerical optimization in the *areas of machine learning, financial mathematics,* and *VLSI design*. A distinctive feature of the book is also to provide basic MATLAB codes as building blocks for readers to develop their own codes for various algorithms discussed in the book.

This text has been designed for *two*, one semester courses of basic study in *numerical optimization with applications* for B.Tech, M.Tech, M.Sc, and M.Phil (Mathematics/Statistics/ Operations Research/ Computer Sciences); and M.B.A students who opt for courses in optimization/operations research. If in certain universities/institutes, there is only *one*, one semester course on optimization/operations research, then appropriate topics can be chosen by the instructor as per his/her requirements. Some universities offer a course on linear programming and game theory at the undergraduate level as well. These topics are also covered in the book. Further, the book can also be used as a reference for researchers and practitioners.

An interesting and useful feature of this book is the e*xamples and end chapter exercises*. Within each chapter, detailed numerical examples and graphical illustrations are included for better understanding of concepts and working of various algorithms. Certain objective type exercises have been specially constructed to check subtle aspects of theory and algorithms. Each chapter ends with a *summary and additional notes* section, which tries to provide some recent references for further study.

Although every care has been taken to make the presentation error free, some errors may still remain and we hold ourselves responsible for that. The readers are requested to kindly communicate errors, if any, at chandras@maths.iitd.ac.in (e-mail address of S. Chandra).

In the long process of writing this book we have been encouraged and helped by many individuals. In particular we would like to thank Professors C. R. Bector, S. K. Gupta, S. C. Duttaroy, N. S. Kambo, R. N. Kaul, T. R. Gulati, S. K. Neogy, Joydeep Dutta,B. Chandra, B. S. Panda, S. Dharmaraja, D. Bhatia, S. R. Arora, S. K. Suneja, C. S. Lalitha and Pankaj Gupta, for their suggestions and interest in this book. We gratefully acknowledge the book grant provided by IIT Delhi and thank the Director, Prof. S. Prasad and the Deputy Director (Faculty) Prof. B. N. Jain for their help in this regard.

A very special thanks and appreciation are due to our Ph.D students Ms. Reshma Khemchandani, Ms. Deepali Gupta, Ms. Anulekha Dhara and Mr. Dhirendra Singh Yadav for their tremendous help during the preparation of the manuscript in LATEX and also reading the manuscript from a student point of view. Ms. Reshma Khemchandani undertook the over all responsibility of preparing and reading the manuscript, drawing figures and writing

MATLAB codes; and without her help it would have been extremely difficult to complete this book.

Last but not the least, we are thankful to Mr. N. K. Mehra and his team of Narosa Publishing House for their help, co-operation and understanding, in publication of the book.

S. Chandra
Jayadeva
A. Mehra

Contents

Preface *vii*

1 Introduction **1**
- 1.1 An Overview of Optimization 1
- 1.2 Optimization Problems 2
- 1.3 Some Simple Illustrative Examples 5

2 Linear Programming **11**
- 2.1 Introduction 11
- 2.2 The Graphical Method and Some Intuitive Results 11
- 2.3 The Simplex Method as an Algebraic Version of The Graphical Method 16
- 2.4 The Simplex Method: Certain Notations 24
- 2.5 Description of the Simplex Method 26
- 2.6 Methods of Artificial Variables 33
- 2.7 Alternate Optima 44
- 2.8 Redundancy in Linear Programming 46
- 2.9 Degeneracy and Cycling 51
- 2.10 The Simplex Tableau in the Condensed Form 53
- 2.11 Summary and Additional Notes 55
- 2.12 Exercises 56

3 Mathematics of the Simplex Method **61**
- 3.1 Introduction 61
- 3.2 Some Basic Definitions 61
- 3.3 Some Elementary Results for LPP 64
- 3.4 The Simplex Algorithm: Main Theorems 70
- 3.5 A Useful Observation 80
- 3.6 The Revised Simplex Method 82
- 3.7 Summary and Additional Notes 93
- 3.8 Exercises 94

4 Duality in Linear Programming **101**
- 4.1 Introduction 101
- 4.2 The Dual Problem: Motivation Through an Example 101
- 4.3 Construction of the Dual 103
- 4.4 Duality Theorems 107
- 4.5 A Convenient Way for Reading the Solution of the Dual 114
- 4.6 Some Useful Deductions from the Complementary Slackness Theorem 118
- 4.7 The Dual Simplex Method 120
- 4.8 Post Optimality Analysis 125
- 4.9 Two Person Zero-Sum Matrix Games 133

4.10 Linear Programming and Matrix Game Equivalence 136
4.11 Summary and Additional Notes 143
4.12 Exercises 144

5 The Transportation and Assignment Problems 151
5.1 Introduction 151
5.2 The Transportation Problem : Description and Mathematical Model 151
5.3 The Balanced Transportation Problem 153
5.4 Consequence of Unimodularity 157
5.5 Optimality Conditions/ Stopping Criterion 161
5.6 Methods for Finding a Starting Solution 162
5.7 The Complete Algorithm 169
5.8 Degeneracy in the Transportation Problem 180
5.9 The Unbalanced Transportation Problem 183
5.10 The Assignment Problem: Description and Mathematical Model 190
5.11 A Transportation Equivalent of the Assignment Problem 194
5.12 An Important Lemma 197
5.13 The Hungarian Method for solving the Assignment Problem 198
5.14 A Dual Interpretation of the Hungarian Method 203
5.15 Finite Convergence of the Hungarian Method 207
5.16 Summary and Additional Notes 208
5.17 Exercises 210

6 Integer Linear Programming 217
6.1 Introduction 217
6.2 Mathematical Model and Some Possible Approaches 217
6.3 Gomory's Cutting Plane Method for All Integer Linear Programming 221
6.4 Gomory's Cutting Plane Method for Mixed Integer Linear Programming 230
6.5 Branch and Bound Method 236
6.6 Some Other Related Problems 241
6.7 Zero-One Implicit Enumeration Algorithm 244
6.8 Summary and Additional Notes 249
6.9 Exercises 249

7 Convex Optimization and Quadratic Programming 255
7.1 Introduction 255
7.2 Convex Functions and their Properties 255
7.3 Convex Optimization Problems 267
7.4 Convex Programming Problems 270
7.5 Optimality Conditions: Motivations from Elementary Calculus 274
7.6 Quadratic Programming 280
7.7 Wolfe's Method for Quadratic Programming 281
7.8 Summary and Additional Notes 292
7.9 Exercises 293

8 Optimality Conditions and Duality in Nonlinear Programming **299**
8.1 Introduction 299
8.2 Feasible Directions and Linearizing Cone 299
8.3 Basic Constraint Qualification 306
8.4 Lagrangian and Lagrange Multipliers 308
8.5 Karush-Kuhn-Tucker Necessary/Sufficient Optimality Conditions 310
8.6 Duality in Nonlinear Programming 314
8.7 Certain Special Cases of Wolfe Dual 317
8.8 Summary and Additional Notes 320
8.9 Exercises 321

9 Unconstrained Optimization Problems **325**
9.1 Introduction 325
9.2 Basic Scheme and Certain Desirable Properties 325
9.3 Line Search Methods for Unimodal Functions 327
9.4 The Steepest Descent Method 336
9.5 Newton's Method 341
9.6 Modified Newton's Method 343
9.7 The Conjugate Gradient Method 344
9.8 Davidon-Fletcher-Powell (DFP) Method 353
9.9 Preconditioning 356
9.10 Summary and Additional Notes 356
9.11 Exercises 357

10 Algorithms in Nonlinear Programming **363**
10.1 Introduction 363
10.2 Frank and Wolfe's Method 364
10.3 Mathematical Justification of Frank and Wolfe's Method 368
10.4 Rosen's Gradient Projection Method 369
10.5 The Penalty Function Method: Motivation 374
10.6 Penalty Function Method: Some Illustrative Examples 378
10.7 Numerical Implementation of the Penalty Function Method 382
10.8 Mathematical Justification of the Penalty Function Method 384
10.9 The Barrier Function Method: Motivation 386
10.10 Analytical Implementation of the Barrier Function Method 387
10.11 Numerical Implementation of the Barrier Function Method 389
10.12 Mathematical Justification of the Barrier Function Method 391
10.13 Starting Point for the Barrier Function Method 392
10.14 Algorithms and Their Global Convergence 393
10.15 Multistage Decision Problems and Dynamic Programming 395
10.16 Summary and Additional Notes 403
10.17 Exercises 405

11 Computational Complexity and Karmarkar's Algorithm for Linear Programming **409**
11.1 Introduction 409
11.2 Simplex Algorithm: A Review of the Basic Facts 410

11.3 Computational Complexity of the Simplex Algorithm: Definitions 411
11.4 Simplex Algorithm is not a Polynomial Time Algorithm 413
11.5 Karmarkar's Algorithm for Linear Programming 419
11.6 Centering Transformation and Projection Matrix 421
11.7 The Complete Algorithm 426
11.8 Putting a general LPP in Karmarkar's form 430
11.9 Worst Case Computational Complexity of Karmarkar's Algorithm 433
11.10 Summary and Additional Notes 435
11.11 Exercises 437

12 Some Generalized Convex Functions and Fractional Programming 441
12.1 Introduction 441
12.2 Quasiconvex and Quasiconcave Functions 441
12.3 Pseudoconvex and Pseudoconcave Functions 454
12.4 Fractional Programming Problems 461
12.5 Linear Fractional Programming Problems 462
12.6 Nonlinear Fractional Programming Problems 468
12.7 Summary and Additional Notes 474
12.8 Exercises 475

13 Multi-objective Optimization: Theory and Methods 479
13.1 Introduction 479
13.2 Conflicting Objectives and Tradeoff 480
13.3 Various Solution Concepts 481
13.4 Weighted Sum Approach 489
13.5 Formulation of Goal Programming Problem 495
13.6 Solution Methodologies for Linear Goal Programming Problems 499
13.7 Summary and Additional Notes 504
13.8 Exercises 505

14 Semi-definite Programming 509
14.1 Introduction 509
14.2 Motivation 509
14.3 Cone 512
14.4 Formulation of Semi-definite Programming Problem 515
14.5 Applications 518
14.6 Formulation of the Dual Problem 523
14.7 Duality Theorems 527
14.8 Summary and Additional Notes 532
14.9 Exercises 533

15 Evolutionary Methods and Global Optimization 535
15.1 Introduction 535
15.2 How Difficult is the Problem? 536
15.3 Simulated Annealing 538
15.4 Genetic Algorithms 542
15.5 Ant Colony Optimization 547
15.6 Summary and Additional Notes 551

16 Mathematical Programming Applications in Machine Learning **553**
16.1 Introduction 553
16.2 Binary (Two-Class) Pattern Classification Problems 554
16.3 Optimal Separation For Linearly Separable Data Sets 560
16.4 Hard Margin Classifier 563
16.5 Soft Margin Classifier 568
16.6 Nonlinear Support Vector Machines 572
16.7 Summary and Additional Notes 576
16.8 Exercises 577

17 Mathematical Programming Applications in Financial Mathematics **579**
17.1 Introduction 579
17.2 Technical Terminologies 579
17.3 Two Asset Portfolio Optimization 583
17.4 Multi Asset Portfolio Optimization 589
17.5 Capital Asset Pricing Model (CAPM) 596
17.6 Summary and Additional Notes 603
17.7 Exercises 605

18 Mathematical Programming Applications in Engineering **609**
18.1 Introduction 609
18.2 Optimization in VLSI Design 609
18.3 Global Routing for VLSI Standard Cells 612
18.4 Wire-Sizing and Buffer Sizing 613
18.5 Synthesis of Antennae Array 616
18.6 Chemical Equilibrium 618
18.7 Protein Folding 619
18.8 Curve Fitting 621
18.9 Reliability Optimization 622
18.10 Wireless Network System 624
18.11 NMR Experiment Design 625
18.12 Summary and Additional Notes 627

19 Matlab Codes For Some Selected Algorithms **629**
19.1 Introduction 629
19.2 Simplex Algorithm 632
19.3 Dual Simplex Algorithm 650
19.4 Assignment Problem: Hungarian Method 655
19.5 Branch and Bound Method 659
19.6 Golden Section Rule 671
19.7 Steepest Descent Method 673
19.8 Conjugate Gradient Method 675
19.9 DFP Method 676
19.10 Frank and Wolfe's Method 678
19.11 General Comments 680

References *681*
Index *691*

1
Introduction

1.1 An Overview of Optimization

Optimization constitutes a very important branch of modern applied mathematics. A variety of problems arising in the areas of engineering design, operations research, management science, computer science, financial engineering and economics can be modeled as optimization problems. There are three major aspects of optimization which are identified as theory, algorithms and applications. It has been possible to use optimization in real life applications because of the availability of efficient algorithms, and these algorithms have been developed because of certain interesting research in optimization theory. Thus, there is very strong linkage between theory and algorithms on one hand, and algorithms and applications on the other hand. Another aspect of optimization which is equally important is the development of good software for various algorithms. Time has proven that the practical value of an algorithm is largely determined by its numerical performance, robustness, and ease of computer implementation. Therefore developing efficient and reliable software for optimization algorithms is certainly very critical.

Though the existence of optimization techniques can possibly be traced to the era of Isac Newton, L. Euler, J. L. Lagrange and A. L. Cauchy, it was the development of the simplex method for linear programming by G. B. Dantzig in the mid 40's which, in a sense, started the subject of mathematical optimization the way we understand it today. Another major development was due to H. W. Kuhn and A. W. Tucker in 1951 who gave necesaary/sufficient optimality conditions for nonlinear optimization or nonlinear programming problems. These conditions are now called as Karush-Kuhn-Tucker (KKT) conditions because W. Karush in 1939 had already developed conditions similar to those given by Kuhn-Tucker.

The algorithmic development in nonlinear optimization got a boost due to the pioneering work by W. C. Davidon, R. Flecther and M. J. D. Powell in late 50's in the area of unconstrained optimization. The sequential unconstrained minimization technique (SUMT) developed by A. V. Fiacco and G. P. McMormick in mid 60's is still considered very efficient for solving nonlinear optimization problems. In 1984, N. Karmarkar developed a polynomial time algorithm for linear programming which not only renewed

attention of many researchers for solving large linear programming problems but also became responsible for the powerful interior point methods for solving structured convex programming problems. The area of applications has also gone through a major change. In addition to the traditional applications in engineering and management, there are areas like machine learning, embedded systems, VLSI, and portfolio management which routinely use latest optimization algorithms.

1.2 Optimization Problems

We start with a generic description of *finite dimensional deterministic optimization problem*. Let f, g_i, $(i = 1,2,\ldots,m)$ and h_j, $(j = 1,2,\ldots,p)$ be functions from $\mathbf{R}^n$ to $\mathbf{R}$ and D be a set in $\mathbf{R}^n$. Then an optimization problem can be stated as

$$\begin{aligned} \text{Min} \quad & f(x) \\ \text{subject to} \quad & \\ & g_i(x) \leq 0 \quad (i = 1,2,\ldots,m) \\ & h_j(x) = 0 \quad (j = 1,2,\ldots,p) \\ & x \in D\,, \end{aligned} \tag{1.1}$$

where $x \in \mathbf{R}^n$ is the vector of unknown decision variables.

A finite dimensional optimization problem as described above is also called a *mathematical programming problem*. Here the function f is called the *objective function* and the functions g_i, $(i = 1,2,\ldots,m)$ and h_j, $(j = 1,2,\ldots,p)$ are called the *constraint functions*. Without any loss of generality, problem (1.1) could also have been stated in the maximization from because Max$(f(x))$=-Min$(-f(x))$.

A point $x \in \mathbf{R}^n$ satisfying all the constraints of problem (1.1) is called a *feasible solution* and the set of all feasible solutions is called the *feasible set* or the *feasible region*, to be denoted by S. If S is empty then the problem is called *infeasible*. If it is possible to find a sequence $\{x^{(k)}\} \in S$ such that $\{f(x^{(k)})\} \to -\infty$ ($\{f(x^{(k)})\} \to +\infty$ for the maximization case), then the problem is called *unbounded*.

A point $\bar{x} \in S$ is called a *global min point* of problem (1.1) if $f(\bar{x}) \leq f(x)$ for all $x \in S$. If $f(\bar{x}) < f(x)$ for all $x \in S$, $x \neq \bar{x}$, then $\bar{x}$ is called a *strict global min point*. If there exists a point $\bar{x} \in S$ such that $f(\bar{x}) \leq f(x)$ for all $x \in S \bigcap N_\delta(\bar{x})$, $N_\delta(\bar{x}) = \{x \in \mathbf{R}^n : \|x-\bar{x}\| < \delta\}$ being the δ−neighborhood of the point $\bar{x}$, then $\bar{x}$ is called a *local min point* of the problem. A *strict local min point* is defined similarly. The definitions of a *global max point*, a *strict global max point*, a *local max point*, and a *strict local max point* can now be given analogously.

Many factors affect whether the given optimization problem can be solved efficiently; e.g. the number of decision variables, the number of constraints and the properties of the objective and constraint functions. We can broadly classify the optimization problem (1.1) into two classes, namely the class of *unconstrained optimization problems*, and

that of *constrained optimization problems*. If $S = \mathbf{R}^n$ then the problem (1.1) is an unconstrained optimization problem, while if $S \subset \mathbf{R}^n$ then it is a constrained optimization problem.

We usually take $D = \mathbf{R}^n$ and do not consider equality constraints separately so as to obtain the following model of an optimization problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \leq 0 \quad (i = 1, 2, \ldots, m). \end{aligned} \tag{1.2}$$

It is often convenient to classify the above optimization problem based on the nature of objective and constraint functions. If all functions f and g_i , $(i = 1, 2, \ldots, m)$ appearing in (1.2), are linear functions of the decision variables x, then it called a *linear optimization problem* or a *linear programming problem (LPP)*. However, if either the objective function f or at least one of the constraints g_i $(i = 1, 2, \ldots, m)$ is nonlinear function of the decision variables x, then it is called a *nonlinear optimization problem* or a *nonlinear programming problem (NLP)*. Since LPP's posses a very rich structure, namely, the structure of linearity, it is expected that they will be easier to solve in comparison to NLP's. For this reason, instead of general purpose optimization algorithms, we develop different algorithms for problems with specific structure. The specific structure generally leads to nice theoretical results which are then exploited for the algorithmic development of the class of optimization problems at hand.

Let us now consider the following optimization problems and classify each of them as a LPP or NLP

(1) $\begin{aligned} &\text{Max} && x_1 - 2x_2 \\ &\text{subject to} \\ & && x_1 + x_2 \leq 4 \\ & && -2x_1 + x_2 \leq 3 \\ & && x_1, x_2 \geq 0. \end{aligned}$

(2) $\begin{aligned} &\text{Min} && |x_1 - 2| + |x_2 - 2| \\ &\text{subject to} \\ & && x_1 - x_2 \geq 0 \\ & && x_1 + x_2 \leq 1 \\ & && x_1, x_2 \geq 0. \end{aligned}$

(3) $\begin{aligned} &\text{Max} && 2x_1 + 4x_2 \\ &\text{subject to} \\ & && x_1^2 + x_2^2 \leq 5 \\ & && x_1 - x_2 \leq 2. \end{aligned}$

(4) Max $(x_1-3)^2+(x_2-2)^2$
subject to
$$x_1^2 - x_2 - 3 \le 0$$
$$-1 \le x_2 \le 1$$
$$x_1 \ge 0.$$

By observing the objective and constraint functions of each of the above problems, we get that (1) is a LPP while (2), (3), and (4) are NLP's.

To have some insight about a possible procedure to solve a general optimization problem and also to appreciate the difficulties involved therein, let us consider problem (4) given above. Here the feasible region consists of all points $(x_1, x_2) \in \mathbf{R}^2$ which satisfy the given constraints. Noting that $x_1^2 = (x_2+3)$ represents a parabola with vertex (0,-3) it is simple to visualize the feasible region as shown in Fig 1.1.

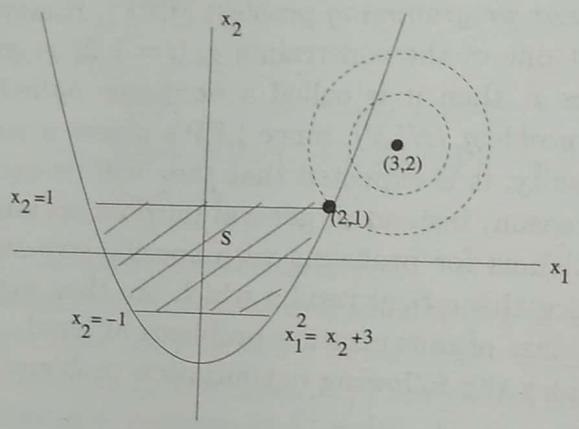


Fig. 1.1.

Now in the feasible region S, we wish to find $(\bar{x}_1, \bar{x}_2)$ for which the value of the objective function $(x_1-3)^2+(x_2-2)^2$ is least. For this, we observe that the contours of the objective function, namely $(x_1-3)^2+(x_2-2)^2 =$ constant, represent a family of circles with center (3,2). Therefore to find the global min point of the given problem, we should find the circle with the smallest radius having center (3,2) which intersects the feasible region S. From Fig. 1.1, it is obvious that such a circle has radius $\sqrt{2}$ which intersects the feasible region at (2,1). Therefore the global min point is $(\bar{x}_1 = 2, \bar{x}_2 = 1)$ and the optimal value is 2.

Remark 1.2.1. The applicability of the above approach is obviously restricted to small problems involving two variables only and therefore it not of much use for general problems. However these geometrical insights help in understanding the general problem and thereby motivate the possible approaches for the algorithmic development.

1.3 Some Simple Illustrative Examples

In this section we present some simple illustrative examples of linear and nonlinear optimization problems. Though these examples could not be labeled as 'realistic', they are good enough to convince us the applicability of numerical optimization. A more detailed discussion on the applications of mathematical programming in areas like *machine learning*, *portfolio optimization* and *engineering design* is presented in Chapters 16, 17, and 18, respectively.

Cargo-Loading Problem

Let there be a vessel which is to be loaded with stocks of N items. Also let each unit of item i have a weight w_i, a volume q_i, and a value v_i for $i = 1, 2, \ldots, N$. The cargo has weight and volume capacity of W and Q respectively. We wish to determine the number of units x_i of each item i $(i = 1, 2, \ldots, N)$ to be loaded on the vessel so that the total value of the cargo is maximum.

The mathematical formulation of this problem is

$$\begin{aligned} &\text{Max} && \textstyle\sum_{i=1}^{N} v_i x_i \\ &\text{subject to} \\ &&& \textstyle\sum_{i=1}^{N} w_i x_i \le W \\ &&& \textstyle\sum_{i=1}^{N} q_i x_i \le Q \\ &&& x_i \ge 0, \quad \text{and integer } (i = 1, 2, \ldots, N). \end{aligned} \tag{1.3}$$

Problem (1.3) is a representative of the general class of optimization problems, known as *knapsack problems*. If we do not have integer constraints on the decision variables x_i, then it becomes a linear programming problem. However if we restrict x_i's to be integer then problem (1.3) becomes an *integer linear programming problem*.

Multi-stage Compressor Problem

In an N-stage unit, an ideal gas has to be compressed from an initial pressure P_0 to a final pressure P_N. Let it be known that for an N-stage unit, the energy E_N be given by

$$E_N = K\left[\left(\frac{P_1}{P_0}\right)^{\alpha} + \left(\frac{P_2}{P_1}\right)^{\alpha} + \ldots + \left(\frac{P_N}{P_{N-1}}\right)^{\alpha} - N\right],$$

where

$K = \dfrac{nRT\nu}{(\nu - 1)} > 0$, P_i =the discharge pressure at i^{th} stage, n =number of moles of gas, R=universal gas constant, T=inlet temperature, $\alpha = \left(\dfrac{\nu - 1}{\nu}\right) > 0$, $\nu = \dfrac{c_p}{c_v}$ = the ratio of specific heats.

The problem is to determine the intermediate discharge pressures $P_1, P_2, \ldots, P_{N-1}$ $(P_0 \le P_1 \le \ldots \le P_{N-1} \le P_N)$ so that the total energy E_N in the process is minimum. If we define $x_i = \dfrac{P_i}{P_{i-1}}$, $(i = 1, 2, \ldots, N)$ and $r = \dfrac{P_N}{P_0}$, then we get the following formulation to minimize E_N,

$$\text{Min} \qquad K \sum_{i=1}^{N} (x_i^{\alpha} - 1)$$

subject to

$$\prod_{i=1}^{N} x_i = r,$$

which is a nonlinear optimization problem.

Cutting Stock Problem

Suppose that metal sheets are produced in rolls of standard fixed length l and standard width w. Further suppose that a customer places a large order for sheets of width w but of varying lengths. Specifically the order requires b_i sheets with length l_i, $(i = 1, 2, \ldots, m)$ and width w. To meet this order, standard sheets are to be cut in a manner so that the demand is satisfied and the wastage is also minimized. If we assume that the scrap pieces are of no use, then the problem boils down to minimize the number of roles needed to satisfy the order.

Now given a standard sheet of length l, there are many ways of cutting it. We call each of these ways as *cutting pattern*, and characterize the jth cutting pattern by the column vector $a^{(j)}$ whose i^{th} component a_{ij} is a non-negative integer denoting the number of sheets of length l_i in the jth pattern. Obviously the vector $a^{(j)}$ represents a cutting pattern if $\sum_{i=1}^{m} a_{ij} l_i \le l$ and each a_{ij} is a non-negative integer. We can take $j = 1, 2, \ldots, n$ because the number of cutting patterns is finite which we denote by n. If we now let x_j denote the number of standard rolls cut according to the jth pattern, then we get the following formulation to minimize the waste and satisfy the order

$$\text{Max} \qquad \sum_{j=1}^{n} x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij}\, x_j \ge b_i \quad (i = 1, 2, \ldots, m)$$

$$x_j \ge 0 \text{ and integer } (j = 1, 2, \ldots, N). \tag{1.4}$$

Problem (1.4) is an integer linear programming problem.

Load Balancing Problem

Let there be n processors and some of which may be already loaded with other work. A new work arrives and we wish to distribute this new work amongst the processors in such a way that the lightest loaded processor has as heavy a load as possible. Let
p_i = current load of the ith processor $(i = 1, 2, \ldots, n)$, $p_i \geq 0$
L = additional total load to be distributed
x_i = fractional of additional load L to be distributed to the ith processor $(i = 1, 2, \ldots, n)$
r =minimum of final loads after the new additional load L has been distributed.

If we assume that the new work load L can be distributed among n processors without increasing any over head, then we get the following load balancing problem

$$\begin{aligned} &\text{Max} && r \\ &\text{subject to} \\ & && r \leq p_i + x_i L \ (i = 1, 2, \ldots, n) \\ & && \sum_{i=1}^{n} x_i = 1 \\ & && x_i \geq 0 \ (i = 1, 2, \ldots, n). \end{aligned} \tag{1.5}$$

Optimization problem (1.5) is a linear programming problem.

Facility Location Problem

Consider the problem of selecting the location of a facility (called supply center) in a city to serve m customers. The supply center supplies a particular commodity e.g. electricity, water or milk, to the customers. The criterion for selecting the location of the supply center is to minimize an appropriate distance function from the center to the customers. Since the supply of goods in the city must be along perpendicular lines (e.g. streets), the most suitable distance function is the rectangular distance function. Therefore if (x_1, x_2) denote the unknown location (coordinates) of the supply center and (a_i, b_i) $(i = 1, 2, \ldots, m)$, be the given (known) location of customer i, then our optimization problem is

$$\underset{(x_1, x_2)}{Min} \ \{ \underset{(1 \leq i \leq m)}{Max} \ (|a_i - x_1| + |b_i - x_2|)\}. \tag{1.6}$$

Now if we write

$$x_0 = \underset{(1 \leq i \leq m)}{Max} \ (|a_i - x_1| + |b_i - x_2|)$$

then problem (1.6) can be written as

$$\begin{aligned} &\text{Max} && x_0 \\ &\text{subject to} \\ & && x_0 - (|a_i - x_1| + |b_i - x_2|) \geq 0 \ (i = 1, 2, \ldots, m), \end{aligned}$$

which is a NLP but can be transformed to a linear programming problem.

Production Planning Problem

Consider a firm which produces a certain product and it wishes to plan its production schedule over a period of time in an optimal manner. Here it is assumed that the demand function is known over the time horizon and this demand must be met. Also it is assumed that the storage cost is proportional to the amount stored, and there is a production cost associated with a given rate of production. To have a mathematical model of this problem we introduce the following notations

$x(t)$ =stock held at time t
$r(t)$ = rate of production at time t
$d(t)$ = demand at time t
$\dot{x}(t) = \dfrac{dx}{dt}$.

Then the production system can be described by the following differential equation

$$\dot{x}(t) = r(t) - d(t), \quad t \in [0, T], \quad x(0) = x_0 \text{ (given)}. \tag{1.7}$$

But (1.7) can equivalently be described by the integral equation

$$x(t) = x(0) + \int_0^t [r(s) - d(s)]ds,$$

with $r(t) \geq 0$, $x(t) \geq 0$ for $t \in [0, T]$.

If we now assume that for the production level r, $c(r)$ is the production cost rate and for the inventory level x, $h(x)$ is the inventory cost rate, then the cost function to be minimized is

$$\int_0^T \Big(c(r(t)) + h(x(t))\Big)\, dt.$$

Therefore we get the following optimization problem

$$\begin{aligned} &\text{Min} && \int_0^T \Big(c(r(t)) + h(x(t))\Big)\, dt \\ &\text{subject to} \\ & && x_0 + \textstyle\int_0^t (r(s) - d(s))ds \geq 0 \quad (0 \leq t \leq T), \\ & && r(t) \geq 0 \qquad\qquad\quad (0 \leq t \leq T). \end{aligned} \tag{1.8}$$

Here the decision vector $r(t)$ which characterizes a product schedule, is an element of the space of all continuous functions on the interval [0,T].

Problem (1.8) is an *infinite dimensional optimization problem* over the space of all continuous functions defined on [0,T], which is an infinite dimensional vector space. Such infinite dimensional optimization problems in a more general framework are studied in *optimal control theory*, which has many applications in engineering, economics and finance. However, as this book is devoted to the study of finite dimensional optimization problems only, the subject of optimal control theory is not discussed here. Apart from the topic of optimal control theory, there are few other important topics in optimization which could also not be included in this book; e.g. *stochastic programming, geometric programming, network flows, and combinatorial optimization.*

2
Lin

2.1

The
gene
lyin
only
effic
hav
rese
solv
ass
stu

sol
in
pr
b

2

2
Linear Programming

2.1 Introduction

The class of linear programming problems (LPP's) constitutes a very important class of general optimization problems. Purely from the mathematical point of view, the underlying structure of *linearity* on LPP's gives many elegant theoretical results which not only characterize the optimal solution but also provide a direction to the development of efficient algorithms for obtaining the same. From the application point of view, LPP's have a long history of being applied to many-many real life problems in operations research/ management science, engineering and economics. Also many algorithms for solving nonlinear programming problems depend on our capability of solving certain associated linear programming problems effectively. Therefore it is imperative that we study LPP's in somewhat greater detail.

This chapter is essentially an elementary introduction of the *simplex method* for solving LPP's. Here the emphasis is on the geometry of the problem that leads to many intuitive results used in the development of the simplex method. These results are not proved here because a more formal mathematical approach of the simplex algorithm is being presented in the next chapter.

2.2 The Graphical Method and Some Intuitive Results

Let us recall from Chapter 1 that an optimization problem is a linear programming problem (LPP) if the objective function as well as all constraints are linear functions of decision variables. Therefore the mathematical model of a linear programming problem is as follows

$$\begin{aligned} &\text{Max(or Min)} \quad z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ &\text{subject to} \\ &\qquad a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \left(\leq, =, \geq\right) b_1 \\ &\qquad a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \left(\leq, =, \geq\right) b_2 \\ &\qquad\qquad \vdots \\ &\qquad a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \left(\leq, =, \geq\right) b_m \\ &\qquad\qquad x_1 \geq 0, \ldots, x_n \geq 0\,. \end{aligned} \tag{2.1}$$

Here it is understood that only one of the three symbols '$\leq$', '$=$', '$\geq$' holds in the first m constraints and these might be different for different constraints. Also we can consider the problem either in the maximization form or in the minimization form, because either form can be converted into the other form by noting that for an arbitrary real valued function f defined over a domain $D \subseteq \mathbf{R}^n$, $\max f(x) = -\min(-f(x))$. In our presentation we shall mostly take the LPP in the maximization form.

To have some idea about a possible procedure for solving LPP's, we consider the following problem in two variables

$$\begin{aligned} &\text{Max} \quad z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &\qquad x_1 + x_2 \leq 8 \\ &\qquad 2x_1 + x_2 \leq 10 \\ &\qquad x_1, x_2 \geq 0. \end{aligned} \tag{2.2}$$

In this linear programming problem, there are only two decision variables so a number of basic facts about a general LPP can possibly be understood geometrically and proved later analytically. For this let us try to identify the set of all point $(x_1, x_2) \in \mathbf{R}^2$ satisfying the constraints of the given LPP. To start with let us take the first constraint $x_1 + x_2 \leq 8$ and note that the line $x_1 + x_2 = 8$ divides $\mathbf{R}^2$ into two half spaces, one for which $x_1 + x_2 \leq 8$ and other for which $x_1 + x_2 \geq 8$. To identify which inequality sign corresponds to which half space, we look the position of a specific point, say $(0, 0)$. As $x_1 + x_2 \leq 8$ holds for $(0, 0)$, all points on or below the line $x_1 + x_2 = 8$ will meet the constraint $x_1 + x_2 \leq 8$ as indicated in Fig 2.1. Similarly we show the half spaces as given by the other constraints as well.

The set of points S satisfying all the constraints is the shaded region in the first quadrant as shown in Fig 2.1. The set S is called the *feasible region* or *feasible set* and points of S are called *feasible solutions*.

Now if we plot the line $4x_1 + 3x_2 = k$ for various values of k, we shall get a family of parallel straight lines and thus know the direction in which the objective function is increasing or decreasing. This direction is depicted by a arrow in Fig 2.1. Here it should

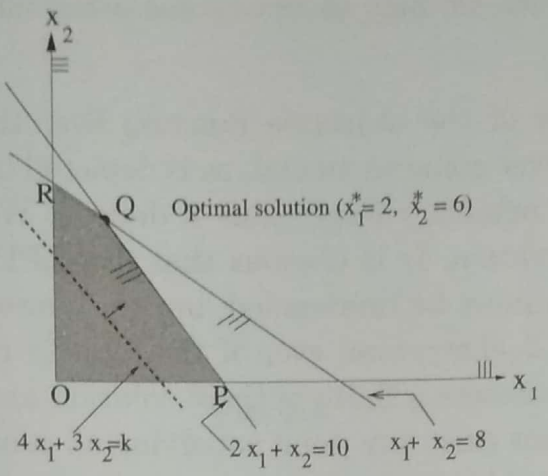


Fig. 2.1.

be noted that because of linearity of the objective function, the gradient is constant and therefore once we know the direction of increase (decrease), in that directions the objective function is going to increase (decrease) for all time to come. This, in general, is not going to happen for nonlinear function as there the gradient depends on the point x, and therefore the function may increase (decrease) for sometime and then start decreasing (increasing). This suggests that for locating the optimal solution of the given LPP, one has simply to slide the line of the objective function over the feasible region and continue sliding till it is possible. In doing so for our example we observe the following simple but important facts

(i) The feasible region is a closed set, which is bounded by straight lines, and has finitely many corner points. Also if we take two points in the feasible region then the whole line segment joining them remains in the feasible region; i.e. it is a *convex set.* Thus the feasible region is a closed, bounded convex set having finitely many corner points. Such a set is called a *polytope.*

(ii) The optimal point to the above problem can not be an interior point. In fact it is a corner point as shown in Fig 2.1.

In view of the above, for locating the maximizing (or minimizing) point of a LPP it looks logical to identify all corner points of the feasible region and choose the one at which the objective function value is maximum (or minimum). This method of solving LPP is called the *graphical method.*

In the context of the example taken here, the four corner points of the feasible region S are $O : (0,0)$, $R : (0,8)$, $P : (5,0)$, and $Q : (2,6)$ and the objective function value is maximum at $(2,6)$ as $z(0,0) = 0$, $z(5,0) = 20$, $z(2,6) = 26$, and $z(0,8) = 24$. Thus $x_1^\star = 2$ and $x_2^\star = 6$ is an optimal solution and $z^\star = 26$ is the optimal value.

While sliding the line of objective function over the feasible region for a general LPP, it may not always happen that we hit only one corner point or sliding will always come to an end. Sometimes, if the feasible region of the LPP is unbounded, or it is empty or

2.3 The Simplex Method as an Algebraic Version of The Graphical Method

In the last section we have observed that the graphical method cannot be applied fo LPP's that have more than two (or at best three) decision variables because one ca not visualize the geometrical object 'corner point' for such problems.

Here it is important to note that this is not a limitation of the method as such, bu rather our inability to visualize geometrical objects beyond three-dimensions. Therefor it seems natural to look into the possibility of understanding the graphical method algebraically. In fact this 'algebraic translation' of the graphical method is possible and that is essentially the well known *simplex method* for linear programming problem developed by G.B. Dantzig in 1947.

To understand the simplex method, we need to understand the concept of a basic feasible solution because as we shall see later, this is precisely the algebraic analogue of the geometrical concept 'corner point'. For this we need to express the given LPP in the standard form by introducing appropriate number of slack and surplus variables at appropriate places. This is done to convert linear inequalities into linear equations so that traditional linear algebra becomes applicable.

In the following we now take an example and describe how the given LPP can be expressed in the standard form. Let the problem be

$$\begin{aligned} \text{Max} \quad & z = 2x_1 + 3x_2 \\ \text{subject to} \quad & \\ & 3x_1 + x_2 \leq 3 \\ & 4x_1 + 3x_2 \geq 6 \\ & x_1 + 2x_2 = 3 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{2.3}$$

Here the first inequality is with '$\leq$' sign so we have to add a non-negative variable, say x_3, so that this can be written as $3x_1 + x_2 + x_3 = 3$.

Similarly the second constraint is with '$\geq$' sign and so we have to subtract a non-negative variable, say x_4, so that this can be written as $4x_1 + 3x_2 - x_4 = 6$.

Variables like x_3 which are nonnegative and are added to '$\leq$' type constraints are called the *slack variables*.

Similarly, variables like x_4 which are nonnegative and are subtracted from '$\geq$' type constraints are called the *surplus variables*.

Now taking the coefficients of the slack and surplus variables as zero in the objective function, we get another linear programming problem as follows

$$\begin{aligned} \text{Max} \quad & z = 2x_1 + 3x_2 + 0x_3 + 0x_4 \\ \text{subject to} \quad & \\ & 3x_1 + x_2 + x_3 = 3 \\ & 4x_1 + 3x_2 - x_4 = 6 \\ & x_1 + 2x_2 = 3 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned} \tag{2.4}$$

There is a very close connection between the linear programming problems (2.3) and (2.4), which we state in the form of below given results.

Result 2.3.1 *There is a one to one correspondence between the feasible solutions of problems (2.3) and (2.4), in the sense that if* (x_1, x_2) *is feasible to (2.3) then there exist unique* x_3 *and* x_4*, namely,* $x_3 = 3 - 3x_1 - x_2$ *and* $x_4 = 4x_1 + 3x_2 - 6$ *so that* (x_1, x_2, x_3, x_4) *is feasible to (2.4); and conversely if* (x_1, x_2, x_3, x_4) *is feasible to (2.4) then* (x_1, x_2) *is feasible to (2.3).*

Result 2.3.2 *Let* $(x_1^\star, x_2^\star)$ *be optimal to (2.3). Then there exist* $x_3^\star = 3 - 3x_1^\star - x_2^\star$ *and* $x_4^\star = 4x_1^\star + 3x_2^\star - 6$ *such that* $(x_1^\star, x_2^\star, x_3^\star, x_4^\star)$ *is optimal to (2.4). Conversely, if* $(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4)$ *is optimal to (2.4) then* $(\bar{x}_1, \bar{x}_2)$ *is optimal to (2.3).*

The proof of the Result 2.3.2 follows from Result 2.3.1 and the fact that in problem (2.4), coefficients of the slack and surplus variables in the objective function are taken as zero. Here it may be remarked that though above results are stated in the context of an example only, it is clear that these hold for a general linear programming problem as well. Some readers may certainly like to prove this last statement themselves.

LPP in Standard Form

In view of Results 2.3.1 and 2.3.2, we can assume without any loss of generality that all constraints of the given LPP are with '=' sign except $(x_j \geq 0, j = 1, 2, \ldots, n)$. Thus any LPP can be taken in the form

$$\begin{aligned} \text{Max} \quad & z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{subject to} \quad & \\ & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ & x_1, \ldots, x_n \geq 0. \end{aligned}$$

If we now introduce vectors x, b, c, and matrix A as

$$x = col(x_1, x_2, \ldots, x_n)$$
$$b = col(b_1, b_2, \ldots, b_m)$$
$$c = col(c_1, c_2, \ldots, c_n)$$
$$A = \begin{bmatrix} a_{11} & a_{11} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}_{m \times n},$$

then the above LPP can be expressed as

$$\begin{aligned} \text{Max} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0 . \end{aligned}$$

Here $x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, $A = (a_{ij}) \in \mathbf{R}^{m \times n}$, and the vector inequality $x \geq 0$ is understand as $x_j \geq 0, j = 1, 2, \ldots, n$.

In the above, we can further assume that $b \geq 0$, because if some component, say b_i, is < 0, then the ith equality can be multiplied by -1 to get $b_i > 0$. Also it can be assumed that Rank $A = m (< n)$, because if this is not true then the system $Ax = b$ is either redundant or has unique solution. Therefore, without any loss of generality we can assume that the given LPP has the following form

$$\begin{aligned} \text{Max} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0, \end{aligned} \tag{2.5}$$

with (i) $b \geq 0$ and (ii) Rank $A = m (< n)$. Any LPP having this form with these two conditions being satisfied is called an *LPP in the standard form.* It is simple to argue that any given LPP can be taken in the standard form without any loss of generality.

If we now refer to problem (2.3) then we note that problem (2.4) is in the standard form with

$$x = col(x_1, x_2, x_3, x_4), \quad c = col(2, 3, 0, 0), \quad b = col(3, 6, 3), \quad \textit{and}$$

$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 4 & 3 & 0 & -1 \\ 1 & 2 & 0 & 0 \end{bmatrix}.$$

Here $b \geq 0$ and Rank $(A) = 3 (< 4)$.

Basic Feasible Solution

Now we proceed to introduce the main concept of this section, namely the *basic feasible solution*, written in short as b.f.s, for the system $Ax = b,\ x \geq 0$ with Rank $A = m(< n)$. Let $a^{(j)}$ $(j = 1, 2, \ldots, n)$ denote the jth column of A, i.e. $a^{(j)} = col(a_{1j}, a_{2j}, \ldots, a_{mj})$ and $A = [a^{(1)}, a^{(2)}, \ldots, a^{(j)}, \ldots, a^{(n)}]$. As Rank $A = m(< n)$, there certainly exists a set of m linearly independent columns of A. We get hold of one such set of m linearly independent columns of A and call these columns as *basic columns*. The remaining $(n - m)$ columns of A are called *non-basic columns*. We next form an $(m \times m)$ matrix B consisting of these m basic columns, and call it a *basis matrix*. Here we may note that there may be more than one basis matrix for the system $Ax = b$ as the choice of m linearly independent columns of A is not unique in general. It is simple to argue that the number of basis matrices is always less than or equal to ${}^{n}C_{m}$.

We shall now define the *basic solution* of the system $Ax = b$ for a given basis matrix B. If the basis matrix B is changed to get a new basis matrix B_1, then the basic solution will also change as the basic solution is defined for a given basis matrix only. Let us now assume that a basis matrix B is given. We note that there is a one to one correspondence between the columns of A and the components of the vector x. For example x_1 corresponds to $a^{(1)}$, x_2 corresponds to $a^{(2)}$ and in general x_j corresponds to $a^{(j)}$. Therefore we can identify those components of vector x which correspond to basis columns, i.e. columns of B. These components of x, which are m in number, are called *basic variables* and remaining $(n - m)$ components of x are called *non-basic variables*. If we now put all $(n - m)$ nonbasic variables as zero and solve the system $Ax = b$ for the m basic variables then the solution so obtained is called the *basic solution* for the given basis matrix B. Some readers may note here that this is a well known result in matrix theory which states that a system of m linear equations in n unknowns, with Rank $A = m(< n)$, has infinitely many solutions depending on $(n - m)$ parameters. For the definition of the basic solution, these $(n - m)$ parameters, namely $(n - m)$ nonbasic variables, are given the zero value.

Next we define a basic feasible solution (b.f.s). For this, we consider the system $Ax = b,\ x \geq 0$. Let B be the given basis matrix. Then we can obtain the basic solution of $Ax = b$ for the given basis matrix B using the procedure as described above. If in this basic solution all basic variables are non-negative then this is called the *basic feasible solution for the basis matrix* B. As all nonbasic variables are zero by definition, a solution is a b.f.s if it is both basic and feasible. For a given basic feasible solution if all basic variables are strictly positive, then it is called a *nondegenerate b.f.s*, otherwise (i.e. when some basic variable takes the value zero) it is called a *degenerate b.f.s*.

Since for the system $Ax = b$ with Rank$(A) = m(< n)$, there will be at most ${}^{n}C_{m}$ basis matrices, the number of basic solutions for the given system is always less than or equal to ${}^{n}C_{m}$. As every basic feasible solution has to be basic solution, the number of basic

feasible solutions for the system $Ax = b,\ x \geq 0$ with $\text{Rank}(A) = m(< n)$ is also less than or equal to ${}^{n}C_{m}$.

Example 2.3.1 *Consider the system*

$$\begin{aligned} x_1 + x_2 + x_3 &= 8 \\ 2x_1 + x_2 + x_4 &= 10 \\ x_1, x_2, x_3, x_4 &\geq 0\,, \end{aligned}$$

and find all basic solutions. Also identify those basic solutions which are basic feasible. Are there any degenerate b.f.s ?

Solution Here we identify

$$x = \quad col(x_1, x_2, x_3, x_4)$$
$$b = col(8, 10)$$
$$\text{and} \quad A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}.$$

We shall first find all possible basis matrices. As Rank $A = 2$, any (2×2) submatrix of A whose determinant is not zero will be a basis matrix. In the matrix A there are six basis matrices and so we shall obtain six basic solutions (one for each basis matrix) for the given system of linear equations. We consider these basis matrices one by one now.

(i) $B_1 = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$. As $|B_1| \neq 0, B_1$ is a basis matrix. Also x_1 and x_2 are basic variables and, x_3 and x_4 are nonbasic variables. Therefore setting $x_3 = 0,\ x_4 = 0$ and solving the resulting system for the basic variables x_1 and x_2 we get $x_1 = 2,\ x_2 = 6$. This gives the basic solution corresponding to the basis matrix B_1 as $(x_1 = 2,\ x_2 = 6,\ x_3 = 0,\ x_4 = 0)$. Further as both basic variables are strictly positive this is a non-degenerate basic feasible solution.

(ii) $B_2 = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$. As $|B_2| \neq 0, B_2$ is a basis matrix. Here x_1 and x_3 are basic variables and, x_2 and x_4 are nonbasic variables. Therefore setting $x_2 = 0,\ x_4 = 0$ we get $x_1 = 5,\ x_3 = 3$. This gives the basic solution corresponding to the basis matrix B_2 as $(x_1 = 5, x_2 = 0, x_3 = 3, x_4 = 0)$ which is again a non-degenerate basic feasible solution.

(iii) $B_3 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$. As $|B_3| \neq 0, B_3$ is a basis matrix. Here x_1 and x_4 are basic variables and, x_2 and x_3 are nonbasic variables. Therefore setting $x_2 = 0,\ x_3 = 0$ we get $x_1 = 8,\ x_4 = -6$. This gives the basic solution corresponding to the basis matrix B_3 as $(x_1 = 8, x_2 = 0, x_3 = 0, x_4 = -6)$ which is not a basic feasible solution as x_4 is negative.

(iv) $B_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$. As $|B_4| \neq 0, B_4$ is a basis matrix. Here x_2 and x_3 are basic variables and, x_1 and x_4 are nonbasic variables. Therefore setting $x_1 = 0,\ x_4 = 0$, we get,

$x_2 = 10,\ x_3 = -2$. This gives the basic solution corresponding to the basis matrix B_4 as $(x_1 = 0, x_2 = 10, x_3 = -2, x_4 = 0)$ which is not a basic feasible solution as x_3 is negative.

(v) $B_5 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. As $|B_5| \neq 0, B_5$ is a basis matrix. Here x_2 and x_4 are basic variables and, x_1 and x_3 are nonbasic variables. Therefore setting $x_1 = 0,\ x_3 = 0$ we get $x_2 = 8,\ x_4 = 2$. This gives the basic solution corresponding to the basis matrix B_5 as $(x_1 = 0, x_2 = 8, x_3 = 0, x_4 = 2)$ which is a non-degenerate basic feasible solution.

(vi) $B_6 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. As $|B_6| \neq 0, B_6$ is a basis matrix. Here x_3 and x_4 are basic variables and, x_1 and x_2 are nonbasic variables. Therefore setting $x_1 = 0,\ x_2 = 0$ we get $x_3 = 8,\ x_4 = 10$. This gives the basic solution corresponding to the basis matrix B_6 as $(x_1 = 0, x_2 = 0, x_3 = 8, x_4 = 10)$ which is a non-degenerate basic feasible solution.

Thus in the above example we have six basic solutions and out of these only four are basic feasible solutions. Further all b.f.s are non-degenerate.

At this stage let us stop for a moment and go back to problem (2.2). In Fig 2.1 we observe that the feasible region has exactly four corner points while our discussion here for the corresponding constraints in the standard form, gives us exactly four basic feasible solutions. Is it just a coincidence or there is something more in this observation? Well, it is not a coincidence. Infact we have obtained the algebraic analogue of the geometrical object corner point in the form of a basic feasible solution, i.e.'corner point' and b.f.s are one and the same. It is just the way of looking them is different, as one is geometric in nature where as the other is algebraic in nature. Also using Result 2.3.1 and Result 2.3.2, we can identify the specific corner point for the given b.f.s. Let us take the b.f.s $(x_1 = 2, x_2 = 6, x_3 = 0, x_4 = 0)$ corresponding to the basis matrix B_1. It corresponds to the corner point (2,6) because to get the feasible point for the given LPP from a feasible point of the corresponding LPP in the standard form, we have to simply ignore the slack and surplus variables. In a similar manner, the b.f.s $(x_1 = 5, x_2 = 0, x_3 = 0, x_4 = 0)$ corresponds to the corner point (5,0). Here it should be noted that this correspondence does not hold for all basic solutions, it holds for b.f.s only. Thus for the basic solution $(x_1 = 8, x_2 = 0, x_3 = 0, x_4 = -6)$ there is no corresponding point in the feasible region of Fig 2.1.

The above discussion can be summarized in the form of following result.

Result 2.3.3 *Every corner point of the feasible set S is a basic feasible solution of the system $Ax = b, x \geq 0$, and conversely every basic feasible solution of the above system is a corner point of the feasible set S.*

Here it should be emphasized that the correspondence between basic feasible solutions and corner points is one to one for non-degenerate b.f.s only, i.e. two non degenerate b.f.s will correspond to two different corner points. But for degenerate b.f.s it is not necessarily true, i.e. more than one degenerate b.f.s may correspond to the same corner

point. This we illustrate with the help of following system

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ x_1 - x_2 + x_4 &= 0 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Here

$$x = \quad col(x_1, x_2, x_3, x_4)$$
$$b = \; col(3, 0)$$
$$\text{and} \quad A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix}.$$

Also, Rank $(A) = 2(< 4)$. Let us now take two basis matrices namely $B_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ and $B_2 = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}$ and find the corresponding basic solutions.

(i) $B_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. For this basis matrix x_3 and x_4 are basic variables, and x_1 and x_2 are nonbasic variables. Therefore taking $x_1 = x_2 = 0$ and solving the resulting system for x_3 and x_4, we get $x_3 = 3$; $x_4 = 0$, thereby getting the basic solution as $(x_1 = 0, x_2 = 0, x_3 = 3, x_4 = 0)$. As all the basic variables are non-negative, this basic solution is a b.f.s. Further, as the basic variable $x_4 = 0$, this is a degenerate b.f.s.

(ii) $B_2 = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}$. For this basis matrix x_2 and x_3 are basic variables, so taking $x_1 = x_4 = 0$, we get $x_2 = 0, x_3 = 3$, thereby getting the basic solution as $(x_1 = 0, x_2 = 0, x_3 = 3, x_4 = 0)$. It is a b.f.s as all the basic variables are non-negative. Further it is a degenerate b.f.s as the basic variable $x_2 = 0$.

The above two degenerate b.f.s are different as B_1 and B_2 are different basis matrices, but they both corresponds to the corner point (0,0,3,0) in $\mathbf{R^4}$ or (0,0) in $\mathbf{R^2}$.

The above discussion motivates us to state the following result.

Result 2.3.4 *Every corner point of the feasible region S is a basic feasible solution of the system $Ax = b$, $x \geq 0$; and conversely every basic feasible solution of the above system is a corner point of the set S. Further for the non-degenerate basic feasible solutions, the correspondence is one to one, i.e. two distinct non-degenerate basic feasible solutions will correspond to two distinct corner points of S, but for the degenerate basic feasible solutions, more than one degenerate basic feasible solutions may correspond to the same corner point.*

Now we can restate Result 2.3.2 replacing corner point by b.f.s and get the following result.

Result 2.3.5 *If the given LPP has an optimal solution then at least one basic feasible solution is optimal.*

In view of Result 2.3.5, a possible way to solve LPP could be to find all b.f.s (at most ${}^nC_m$) and choose the one at which the objective function value is maximum. An important point to note here is that the main difficulty of the graphical method, namely corner points can not be computed and visualized beyond two or three dimensions, is no more there. This is because theoretically, all b.f.s can be computed, no matter how many variables are there or how many constraints are there.

Though the above logic seems to convince us that now we have an algorithm which is capable of solving every linear programming problem. But one can imagine how much time consuming it will be to find all ${}^nC_m$ (in the worst case) b.f.s and then choose the one at which the objective function value is maximum; and therefore this can not be taken as an appropriate method to solve LPP's. Nevertheless, corner points (or b.f.s) are important and we have to concentrate on them to develop an efficient algorithm for solving LPP's. A more practical approach seems to be the one which does not need all b.f.s at one go, rather starts from an initial b.f.s only. Then it verifies (using some mathematical and geometrical arguments due to the structure of linearity) if the current b.f.s is optimal. If *NOT*, then in a systematic logical manner it generates a new b.f.s, which is an improved one, and this process is continued till a suitable optimality criteria is satisfied. This is what precisely the *simplex method* does. Geometrically it checks, the optimality of current b.f.s (i.e. the current corner point) by verifying if the value of the objective function at the given corner point is more in comparison with the value of the objective function at all other corner points which are joined by an edge, i.e. all adjacent corner points. If this happens, then it will mean that current corner point (equivalently the current b.f.s) is a local maximum point, which because of linearity will assert that it is a global maximum point. Here it may be noted that even *adjacent corner points* are not known explicitly, so desired information is achieved implicitly. However if the current b.f.s is not optimal then the simplex method generates a new b.f.s so that the objective function value increases (it does not decrease). In fact it generates such a b.f.s by giving a corner point that is adjacent to the current corner point. Thus the simplex method moves along the adjacent corner points only, i.e. it moves from the current corner point to another corner point joined by an edge so that the objective function value improves. Since the number of corner points are finite (at most ${}^nC_m$), the method will terminate in finite number of iterations (except in certain rare situations called 'cycling' to be discussed later). We now continue our discussion on the simplex method in a somewhat detailed manner.

2.4 The Simplex Method: Certain Notations

We now consider the standard form LPP and with respect to that introduce the following notations to be used subsequently

$a^{(j)} = col(a_{1j}, a_{2j}, \ldots, a_{mj})$;
$A = [a^{(1)}, a^{(2)}, \ldots, a^{(j)}, \ldots, a^{(n)}]$
$B = (b^{(1)}, b^{(2)}, \ldots, b^{(m)})$
$x_B = B^{-1}b = col(x_{B1}, x_{B2}, \ldots, x_{Bm})$
$y^{(j)} = col(y_{1j}, y_{2j}, \ldots, y_{ij}, \ldots, y_{mj}] = B^{-1}a^{(j)}, (j = 1, 2, \ldots, n)$
$c_B = col(c_{B_1}, c_{B_2}, \ldots, c_{B_i} \ldots, c_{B_m})$,

c_{B_i} being the coefficient of basic variable $x_{B_i}, (i = 1, 2, \ldots, m)$ in the objective function, $z(x_B) = c_B^T x_B = \sum_{i=1}^{m} c_{B_i} x_{B_i}$; and $z_j = c_B^T y^{(j)} (j = 1, 2, \ldots, n)$.

It is not difficult to understand what above notations really mean. We know that A is an $(m \times n)$ coefficient matrix and therefore if by $a^{(j)}$ we denote the jth column $(j = 1, 2, \ldots, n)$ of A then we can write $A = [a^{(1)}, a^{(2)}, \ldots, a^{(j)}, \ldots, a^{(n)}]$. Also as Rank$(A) = m (< n)$, there exist m linearly independent columns in A which we have denoted by $b^{(1)}, b^{(2)}, \ldots, b^{(m)}$ and called them as basic columns. Here it may be noted that these are not new columns but come from the matrix A itself, i.e. $b^{(1)}$ is nothing but one of the columns of A, and similarly other columns $b^{(2)}, \ldots, b^{(m)}$ as well. Since we do not know the exact indices of these basic columns in general, we have used a different notation for these. Further B is an $(m \times m)$ matrix consisting of these m basic columns, it is a basis matrix and so invertible. The $(m \times 1)$ vector $x_B = B^{-1}b$ gives the values of m basic variables $x_{B1}, x_{B2}, \ldots, x_{Bm}$. Again this is not a new vector; its components are m components of x which correspond to basic variables. If by x_R we denote the vector of non-basic variables then $(x_B = B^{-1}b,\ x_R = 0)$ is the basic solution for the basis matrix B. In case all components of the vector $B^{-1}b$ are non-negative, it is a basic feasible solution. As non basic variables are always zero, we shall call $x_B = B^{-1}b$ itself a basic solution or a basic feasible solution, as the case may be. The m basic columns of A form a basis for $\mathbf{R}^m$ and so any column $a^{(j)}$ of A can be written as a linear combination of these basic columns i.e. $a^{(j)} = y_{1j}b_1 + y_{2j}b_2 + \ldots + y_{mj}b_m$ giving $a^{(j)} = By^{(j)}$ or $y^{(j)} = B^{-1}a^{(j)}$. Thus the m elements of the vector $y^{(j)}$ give the values of the coefficients which appear when a nonbasic column $a^{(j)}$ is expressed as a linear combination of m basic columns. As c_B denotes the coefficient of m basic variables in the objective function and $x_R = 0$, $z(x_B) = c_B^T x_B$ gives the value of the objective function for the b.f.s $x_B = B^{-1}b$. The scalars $(z_j - c_j)$ with $z_j = c_B^T y^{(j)}$ are called *relative cost coefficients* (or dual variables) and their sign will determine if making a nonbasic variable x_j positive will improve the objective function or not.

Let us explain the above notations with regard to the following LPP

$$\text{Max} \quad z = 4x_1 + 3x_2$$

subject to

$$x_1 + x_2 \leq 8$$
$$2x_1 + x_2 \leq 10$$
$$x_1, x_2 \geq 0.$$

After adding the slack and surplus variables we get

$$\text{Max} \quad z = 4x_1 + 3x_2 + 0x_3 + 0x_4$$

subject to

$$x_1 + x_2 + x_3 = 8$$
$$2x_1 + x_2 + x_4 = 10$$
$$x_1, x_2, x_3, x_4 \geq 0.$$

Therefore

$$x = col(x_1, x_2, x_3, x_4)$$
$$b = col(8, 10)$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix},$$

and Rank $(A) = 2(< 4)$.

Now if we take $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, as the starting basis matrix then $B = B^{-1} = I$ and so

$$x_B = \begin{pmatrix} x_{B1} = x_3 \\ x_{B2} = x_4 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 8 \\ 10 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$

$$y^{(1)} = \begin{pmatrix} y_{11} \\ y_{21} \end{pmatrix} = B^{-1}a_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$y^{(2)} = \begin{pmatrix} y_{12} \\ y_{22} \end{pmatrix} = B^{-1}a_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$y^{(3)} = \begin{pmatrix} y_{13} \\ y_{23} \end{pmatrix} = B^{-1}a_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$y^{(4)} = \begin{pmatrix} y_{14} \\ y_{24} \end{pmatrix} = B^{-1}a_4 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$c_B = \begin{pmatrix} c_{B1} = c_3 \\ c_{B2} = c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$
\begin{aligned}
z(x_B) &= c_B^T x_B = 0 \\
z_1 &= c_B^T y^{(1)} = 0 \\
z_2 &= c_B^T y^{(2)} = 0 \\
z_3 &= c_B^T y^{(3)} = 0 \\
z_4 &= c_B^T y^{(4)} = 0.
\end{aligned}
$$

$z_1 - c_1 = 0 - 4 = -4$
$z_2 - c_2 = 0 - 3 = -3$
$z_3 - c_3 = 0 - 0 = 0$
$z_4 - c_4 = 0 - 0 = 0.$

Now whatever quantities we have calculated with respect to the current basis matrix $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, we arrange them in the form of a table called *simplex tableau*. Here the simplex tableau is

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 8$	1	1	1	0
$x_4 = 10$	2	1	0	1
$z(x_B) = 0$	−4	−3	0	0

In general the simplex tableau for a given basis matrix B will look like

x_B	$y^{(1)}$	$y^{(2)}$	...	$y^{(j)}$	...	$y^{(n)}$
$x_{B1} = (B^{-1}b)_1$	y_{11}	y_{12}	...	y_{1j}	...	y_{1n}
$x_{B2} = (B^{-1}b)_2$	y_{21}	y_{22}	...	y_{2j}	...	y_{2n}
⋮						
$x_{Bi} = (B^{-1}b)_i$	y_{i1}	y_{i2}	...	y_{ij}	...	y_{in}
⋮						
$x_{Bm} = (B^{-1}b)_m$	y_{m1}	y_{m2}	...	y_{mj}	...	y_{mn}
$z(x_B)$	$(z_1 - c_1)$	$(z_2 - c_2)$	...	$(z_j - c_j)$	...	$(z_n - c_n)$

Here the values of x_{Bi} and $z(x_B)$ are $x_{Bi} = (B^{-1}b)_i$ $(i = 1, 2, \ldots, m)$ and $z(x_B) = c_B^T x_B$ respectively; $(B^{-1}b)_i$ being the ith component of the vector $B^{-1}b$.

2.5 Description of the Simplex Method

Let the given LPP be in the standard form, i.e. it is in the form (2.5). Then following are the five basic steps in the simplex method and we shall detail each of these in this sequel. Here we assume that the given LPP is feasible. Later we shall see that this assumption itself will be verified by the simplex method.

Step 1 Start with an initial basis matrix B such that the basic solution $x_B = B^{-1}b$ is

basic feasible i.e. all $x_{Bi} \geq 0, (i = 1, 2, \ldots, m)$.

Step 2 Check if the b.f.s at hand is optimal. If yes, then we stop.

Step 3 Check if the given LPP has unbounded solution. If yes, then we again stop.

Step 4 If the current b.f.s is not optimal and there is no indications of unbounded solution then generate another b.f.s $\widehat{x_B}$ such that $z(\widehat{x_B}) \geq z(x_B)$.

Step 5 Continue above steps till we obtain an optimal solution or there is an indication of unbounded solution.

Now we discuss each of the above steps one by one. Let us first discuss how to obtain initial b.f.s. For this we note that if all constraints are with '$\leq$' sign then this step is trivial. We simply get hold of m slack columns to have an $(m \times m)$ identity matrix as starting basis matrix and obtain the corresponding initial b.f.s $x_B = B^{-1}b = I\,b = b\,(\geq 0)$. In the case of mixed constraints, we use the *method of artificial variables* for obtaining the initial b.f.s which we plan to discuss later. This latter method will also verify if the given LPP is feasible or infeasible.

We shall now assume that we have a starting b.f.s, say $x_B = B^{-1}b$. For this b.f.s we form the simplex tableau as described earlier. Then we have the following important results whose proofs we shall have in the next chapter. These results are stated with regard to the current simplex tableau and they help us in performing Steps 2, 3 and 4 as described above.

Result 2.5.1 *If all* $(z_j - c_j) \geq 0$ *then the current b.f.s is optimal*

Result 2.5.2 *If some* $(z_j - c_j) < 0$ *and corresponding to that all* $y_{ij} \leq 0$ *then the given LPP has unbounded solution.*

Result 2.5.3 *If some* $(z_j - c_j) < 0$ *and for that some* $y_{ij} > 0$ *then there exists a new b.f.s* $\widehat{x_B}$ *such that* $z(\widehat{x_B}) \geq z(x_B)$.

So once the given LPP has finite optimum and the current b.f.s x_B is not optimal, Result 2.5.3 above guarantees the existence of a new b.f.s $\widehat{x_B}$ such that $z(\widehat{x_B}) \geq z(x_B)$. In the absence of degeneracy $z(\widehat{x_B}) > z(x_B)$ holds.

As per the development of the simplex method, the new b.f.s $\widehat{x_B}$ is obtained by taking one column out of B and entering another column of A which is not already a basic column and thereby getting the new basis matrix $\widehat{B}$ such that $z(\widehat{x_B}) \geq z(x_B)$. Geometrically this means moving from the current corner point along an edge to get an improved adjacent corner point.

The obvious question here is how to decide which column $a^{(j)}$ of A should be entered in B and which column $b^{(r)}$ of B should be taken out. For this we should concentrate on our three basic goals. These are (i) the new matrix $\widehat{B}$ is a basis matrix so that the new solution $\widehat{x_B}$ is a basic solution (ii) every component of $\widehat{x_B}$ is non-negative so that it is a b.f.s and (iii) the objective function value improves (at least it remains at the same level) i.e. $z(\widehat{x_B}) \geq z(x_B)$. As the proof of Result 2.5.3 is constructive, there is a proper procedure to choose indices j and r so that the above three stated goals are met. We

describe these details under the heading 'Rule for a column to enter' and 'Rule for a column to leave'.

Rule for a column to enter

We choose any j for which the hypothesis of Result 2.5.3 hold i.e. $(z_j - c_j) < 0$ and for that some $y_{ij} > 0$. However, in practise, we choose negative most such $(z_j - c_j)$ with the *hope* that we *may* get maximum improvement at this stage. Therefore, if $(z_k - c_k) = \underset{j}{Min}\{(z_j - c_j) : (z_j - c_j) < 0, \text{and some } \ y_{ij} > 0\}$ i.e. $(z_k - c_k)$ is the negative most value of $(z_j - c_j)$ for which at least one $y_{ij} > 0$, then column $a^{(k)}$ enters the basis and the corresponding variable x_k becomes a basic variable.

Rule for a column to leave the basis

Once the column to enter, namely $a^{(k)}$, has been identified we determine the column to leave the basis. For this let

$$\frac{x_{Br}}{y_{rk}} = \underset{i}{Min}\left\{\frac{x_{Bi}}{y_{ik}} : y_{ik} > 0\right\}.$$

Then column $b^{(r)}$ leaves the basis and the corresponding variable x_{Br} becomes a nonbasic variable. In case the minimum ratio is attained for more than one value of r (say r_1 and r_2) then any one of these can be chosen, i.e. either x_{Br_1} or x_{Br_2} is made a non basic variable in the next iteration.

After getting the new basis matrix, say $\widehat{B}$, we compute all quantities x_B, y_j, c_B, $z(x_B)$, $(z_j - c_j)$ etc. a fresh and repeat the whole procedure.

This process has to be continued till we get an optimal solution (using Result 2.5.1) or there is an indication that the given LPP has unbounded solution (using Result 2.5.2).

Since the number of basic feasible solutions is finite (at most ${}^nC_m$), in the absence of degeneracy, the whole procedure has to be finite. This is because two consecutive non-degenerate b.f.s x_B and $\widehat{x_B}$ will correspond to two distinct corner points and $z(\widehat{x_B}) > z(x_B)$. Since more than one degenerate b.f.s may correspond to the same corner point, in a rare situation, there may be some pathological situations and those we plan to discuss later.

The flow chart given in Fig. 2.6 details all the steps involved in the simplex method.

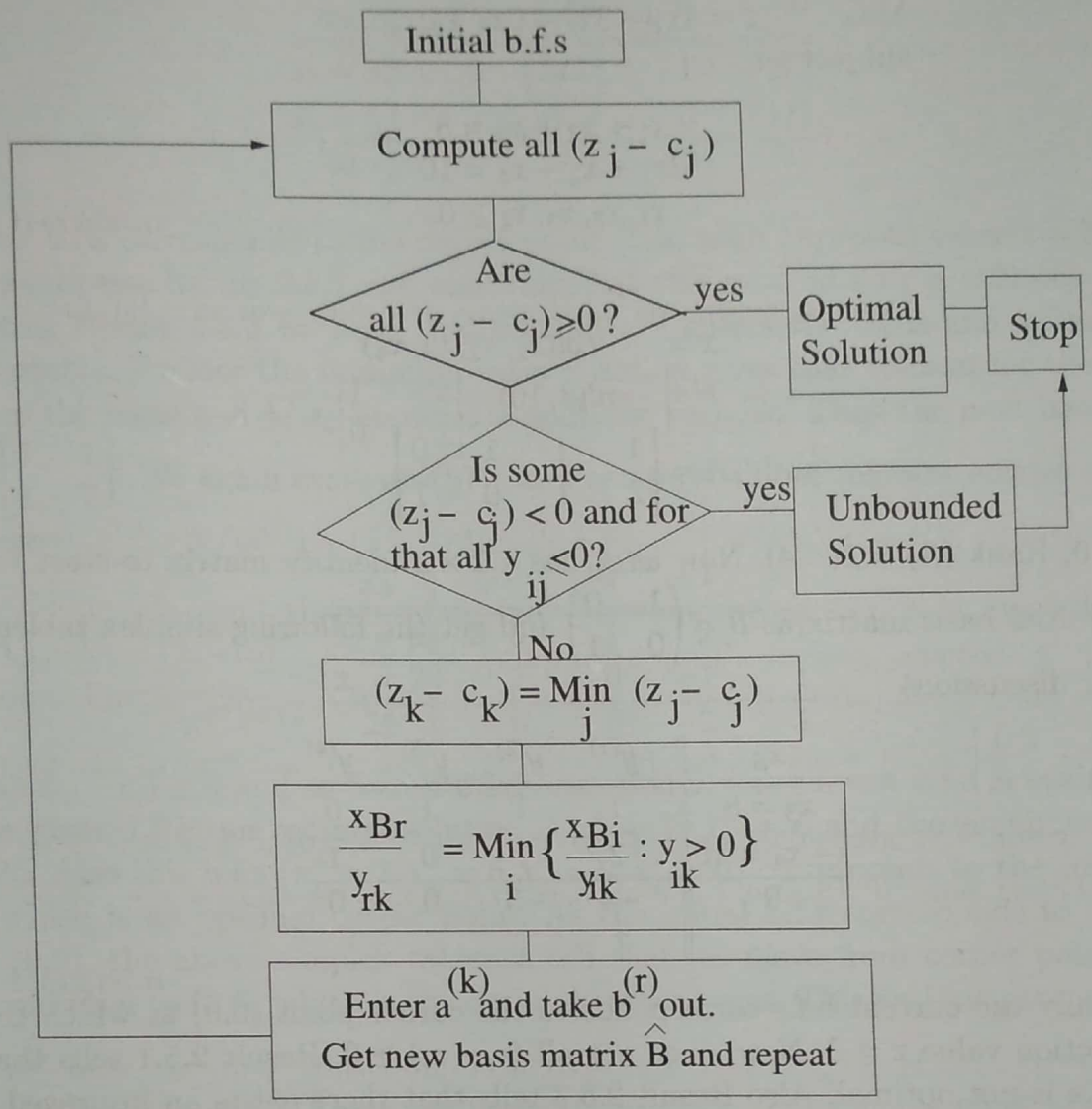


Fig. 2.6.

Example 2.5.1 *Use the simplex method to solve the following LPP and identify the movements graphically*

$$\text{Max} \quad z = 4x_1 + 3x_2$$

subject to

$$x_1 + x_2 \leq 8$$
$$2x_1 + x_2 \leq 10$$
$$x_1, x_2 \geq 0.$$

Solution We have already solved this LPP graphically and obtained its optimal solution as $x_1^\star = 2$, $x_2^\star = 6$ with the optimal value $z^\star = 26$. Let us now solve this problem by the simplex method. For this we have to first express it in the standard form by adding slack and surplus variables as required. This gives the following

$$\text{Max} \quad z = 4x_1 + 3x_2 + 0x_3 + 0x_4$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 8 \\ 2x_1 + x_2 + x_4 &= 10 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Here

$$x = \quad col(x_1, x_2, x_3, x_4)$$
$$b = \quad col(8, 10)$$
$$\text{and} \quad A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}.$$

Also $b \geq 0$, Rank $(A) = 2(< 4)$. Now as A has a 2×2 identity matrix to start with we take the initial basis matrix as $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and get the following simplex tableau (see our earlier discussion)

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 8$	1	1	1	0
$\leftarrow x_4 = 10$	$\boxed{2}$	1	0	1
0	−4	−3	0	0
	↑			

Geometrically the current b.f.s corresponds to the corner point (0,0) at which the objective function value $z = 0$. Now since not all $(z_j - c_j) \geq 0$, Result 2.5.1 tells that the current b.f.s is not optimal. Also Result 2.5.3 tells that there exists an improved b.f.s. For finding out that, we follow the above mentioned rules for entering a column $a^{(k)}$ and taking a column $b^{(r)}$ out of B. As there are two negative values of $(z_j - c_j)$, namely $(z_1 - c_1) = -4$ and $(z_2 - c_2) = -3$ and for each of them some $y_{ij} > 0$ exists, we choose the negative most value of such $(z_j - c_j)$, i.e. $(z_1 - c_1) = -4$. This corresponds to $j = 1$ and so first column $a^{(1)}$ of A enters the basis i.e. x_1 becomes a basic variable. Next we have to decide which column should go out of B. For this we find the ratio $\left\{\frac{x_{Bi}}{y_{ik}} : y_{ik} > 0\right\}$, i.e. $\left(\frac{8}{1}, \frac{10}{2}\right)$ and choose the minimum. Here minimum is 5 which corresponds to the variable x_4. So x_4 becomes a nonbasic variable and the corresponding column $b^{(2)} = a^{(4)}$ goes out of B. Therefore the new basis matrix is $\widehat{B} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}$.

As we have followed the rules prescribed as per the proof of Result 2.5.3, it is guaranteed that $\widehat{B}$ will give an improved b.f.s $\widehat{x_B}$.

Now it can be verified that $(\widehat{B})^{-1} = \begin{pmatrix} 1 & -1/2 \\ 0 & 1/2 \end{pmatrix}$ and therefore computing the new values of x_B, $y^{(j)}$, c_B, $z(x_B)$, $(z_j - c_j)$ we get the following simplex tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 3$	0	$\boxed{1/2}$	1	$-1/2$
$x_1 = 5$	1	1/2	0	1/2
20	0	-1	0	2

This new b.f.s corresponds to the corner point (5,0) with improved value $z = 20$.

We again use Result 2.5.1 and conclude that this new b.f.s $\widehat{x_B}$ is still not optimal. Also using Result 2.5.3 we note that column $a^{(2)}$ enters the basis and x_2 becomes a basic variable. Further the minimum ratio criterion gives that column for the variable x_3 leaves the basis and so x_3 becomes a nonbasic variable. Thus the next basis matrix is $\widehat{\widehat{B}} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$. We again evaluate $(\widehat{\widehat{B}})^{-1}$ and compute all the relevant entries to get the following

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 6$	0	1	2	-1
$x_1 = 2$	1	0	-1	1
26	0	0	2	1

Now all $(z_j - c_j) \geq 0$ and so Result 2.5.1 asserts that the current b.f.s is optimal. Thus for the given LPP, an optimal solution is $x_1^\star = 2$, $x_2^\star = 6$ and the maximum value is $z^\star = 26$. Also this b.f.s $(x_1^\star = 2, x_2^\star = 6, x_3^\star = 0, x_4^\star = 0)$ corresponds to the corner point (2,6) which is an optimal corner point. As the initial b.f.s corresponds to the corner point (0,0), the above simplex tableaus tell that we move from corner point (0,0) to (5,0) and then to (2,6) which is the optimal corner point. This we illustrate in Fig. 2.7.

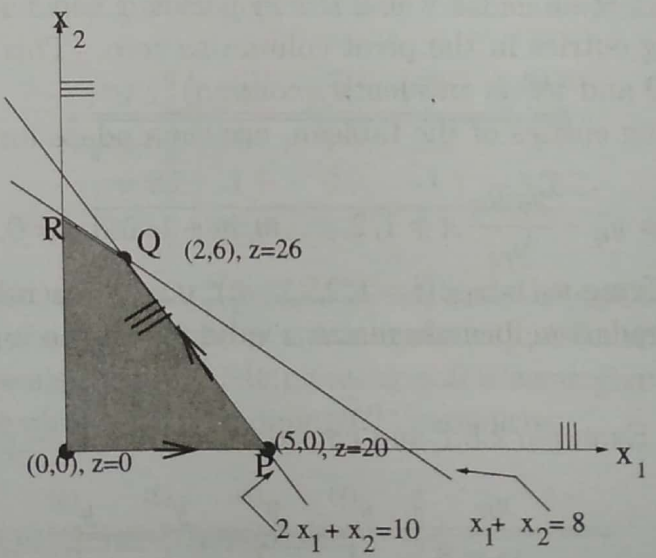


Fig. 2.7.

If we now have a second look at the above solution procedure to solve Example 2.5.1, we note that there is something which is still not very satisfactory, because to get the new elements in the simplex tableau we are first computing the inverse of the new basis matrix $\widehat{B}$ explicitly, and then using the same to get $x_B, y^{(j)}, z_j$ etc. The obvious question is that can we do something better so that we do not compute $(\widehat{B})^{-1}$ explicitly. The answer is yes and that leads us to what is called *pivoting in the simplex method*.

Pivoting In the Simplex Method

In the simplex method, we use *pivoting* to obtain new entries in the simplex tableau (corresponding to the new basis matrix $\widehat{B}$) without finding $(\widehat{B})^{-1}$ explicitly. The basic steps in the pivoting are as follows

Step 1 Identify the *pivot column*. This corresponds to the column for the entering variable in the tableau.

Step 2 Identify the *pivot row*. This corresponds to the row for the leaving variable in the tableau as obtained by the minimum ratio criteria.

Step 3 Identify the *pivot element*. This is the element, which is common to both pivot column and pivot row. Let this be denoted by y_{pq}, where in the simplex tableau p^{th} row is the pivot row and the qth column is the pivot column.

Step 4

(i) Divide all entries in the pivot row by the pivot element, i.e.

$$\widehat{y_{pj}} = \frac{y_{pj}}{y_{pq}}, \quad (j = 0, 1, 2, \ldots, n), \text{ where } y_{p0} \equiv x_{Bp}.$$

This will give $\widehat{y_{pq}} = 1$.

(ii) Take all remaining entries in the pivot column as zero. (This is because for a basic variable $z_j - c_j = 0$ and $y^{(j)}$ is an identity column)

(iii) For the remaining entries of the tableau, use the update formula

$$\widehat{y_{ij}} = y_{ij} - \frac{y_{pj} y_{iq}}{y_{pq}}, i = 1, 2, \ldots, m, m+1 \text{ and } j = 0, 1, 2, \ldots, n.$$

Here we are identifying $y_{i0} = x_{B_i}, (i = 1, 2, \ldots, m)$, $y_{m+1,0} = z(x_B)$ and $y_{m+1,j} = (z_j - c_j)$, so that the same updation formula remains valid for all the remaining entries of the tableau.

Let us now revisit Example 2.5.1 and consider the initial simplex tableau

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_3 = 8$	1	1	1	0
$\leftarrow$	$x_4 = 10$	$\boxed{2}$	1	0	1
	0	−4	−3	0	0
		$\uparrow$			

for the initial basis matrix $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. As discussed in Example 2.5.1 the column $a^{(1)}$ enters the basis (so that in the new tableau x_1 becomes a basic variable) and column $b^{(2)} = a^{(4)}$ leaves the basis (so that in the new tableau x_4 becomes a nonbasic variable) and therefore the new basis matrix is $\widehat{B} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}$. Now we identify pivot column, pivot row and pivot element in the above tableau as indicated and perform Step 4 of the pivoting. This gives

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$\leftarrow x_2 = 3$	0	$\boxed{1/2}$	1	$-1/2$
$x_1 = 5$	1	1/2	0	1/2
20	0	-1	0	2
		$\uparrow$		

Here the second row is obtained by dividing the pivot row by the pivot element $y_{pq} = 2$, and then taking other entries in the corresponding column as zero. This gives the second row and the second column in the new tableau. For the remaining elements we have to use the updation formula as described in step 4. For example the new value of x_3, namely $\widehat{x_3}$ is obtained as $\widehat{x_3} = 8 - \frac{1\times 10}{2} = 8 - 5 = 3$.
Similarly, if $\widehat{z}$ and $\widehat{z_2} - c_2$ respectively denote the new values of z and $z_2 - c_2$, then

$$\widehat{z} = z(\widehat{x_B}) = 0 - \frac{(-4)(10)}{2} = 20$$

$$\widehat{z_2} - c_2 = -3 - \frac{(-4)(1)}{2} = -3 + 2 = -1 \text{ etc.}$$

In a similar manner using pivoting in the new tableau so obtained, we get the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 6$	0	1	2	-1
$x_1 = 2$	1	0	-1	$-3/2$
26	0	0	2	1

which gives the optimal solution $x_1^\star = 2$, $x_2^\star = 6$ and the optimal value $z^\star = 26$ as obtained already. We shall give proper mathematical justification of pivoting in the next chapter where we shall prove that pivoting will always give the same tableau as the one we would have obtained by finding $(\widehat{B^{-1}})$ explicitly.

2.6 Methods of Artificial Variables

While discussing Step 1 of the simplex method, we have agreed that if the coefficient matrix A has an $(m \times m)$ identity matrix then that should be taken as the initial basis matrix to give $x_B = B^{-1}b = I\ b = b(\geq 0)$ as the initial b.f.s. However in case the

constraints of the given LPP are mixed type, we shall in general, not have a $(m \times m)$ identity matrix to start with. As we have already seen the advantage of starting with a $(m \times m)$ identity matrix (virtually no computation is required for the initial simplex tableau) we are tempted to introduce (artificially or as per force) as many identity columns as are required to have an $(m \times m)$ identity matrix. Since the given LPP is in the standard form, all constraints are already with '=' sign, no new variables can be added at a positive level. But unless a new variable is added in a constraint there is no possibility of introducing an identity column at that place. So we introduce these variables 'artificially', i.e. wherever identity column is missing say i^{th} constraint, we introduce a variable x_{a_i} and call that as the ith artificial variable. Here it should be noted that if the constraints of the given LPP are consistent, i.e. the given LPP is feasible, the variable x_{a_i} can not take positive value. Therefore, we take appropriate precautions to make these artificial variables zero as early as possible. Further as long as some $x_{a_i} > 0$, geometrically speaking we are not occupying a feasible corner and are somewhere outside the feasible region. It is only when all $x_{a_i} = 0$ then a feasible corner is obtained. If in some problem it is not possible to make all $x_{a_i} = 0$ eventually then that LPP should be infeasible or equivalently the constraints are inconsistent.

At this stage we may think that why can't we start with, a non-identity matrix. Theoretically we can certainly do it but for any meaningful large problem (never think that we are interested in LPP's of 2 or 3 variables only-there may be hundreds or thousands of variables) it is almost an impossible task because we may have to check all possible ${}^{n}C_{m}$ combinations and then check that we really get a basis matrix which gives rise to a b.f.s.

There are two popular methods which require the introduction of artificial variables to give an initial b.f.s. These are called the *two phase method* and the *big-M method.* We now explain each of these through examples only.

The Two Phase Method

We shall explain the *two phase method* with the help of the following LPP

$$\begin{aligned} \text{Min} \quad & z = 2x_1 + x_2 \\ \text{subject to} \quad & \\ & 3x_1 + x_2 = 3 \\ & 4x_1 + 3x_2 \geq 6 \\ & x_1 + 2x_2 \leq 3 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{2.6}$$

After converting it in the standard form we get the following

$$\text{Max } z = -2x_1 - x_2 + 0x_3 + 0x_4$$

subject to

$$\begin{aligned} 3x_1 + x_2 &= 3 \\ 4x_1 + 3x_2 - x_3 &= 6 \\ x_1 + 2x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned} \tag{2.7}$$

Here the matrix $A = \begin{bmatrix} 3 & 1 & 0 & 0 \\ 4 & 3 & -1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$ does not have a (3×3) identity matrix. Infact we are missing the columns $(1,0,0)$ and $(0,1,0)$ as the column $(0,0,1)$ is already present due to the slack variables x_4. So we add two artificial variables x_{a_1} and x_{a_2} (both ≥ 0) and construct the following Phase I problem

$$\text{Max } z_a = -x_{a_1} - x_{a_2}$$

subject to

$$\begin{aligned} 3x_1 + x_2 + x_{a_1} &= 3 \\ 4x_1 + 3x_2 - x_3 + x_{a_2} &= 6 \\ x_1 + 2x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4, x_{a_1}, x_{a_2} &\geq 0. \end{aligned} \tag{2.8}$$

Here we note that the coefficient matrix of the Phase-I problem (2.8) has desired (3×3) identity matrix so starting solution for solving the Phase-I problem by the simplex method is readily available.

Since all $x_{ai} \geq 0$, the objective function value of the Phase-I problem is ≤ 0. Hence if constraints of the given LPP are consistent the optimal value of the Phase-I problem must be zero.

Thus after solving the Phase-I problem by the usual simplex method, there are two possibilities which may arise. The first possibility is that the optimal value of the Phase-I objective function is zero or equivalently in the optimal solution of the Phase-I problem, all artificial variables are appearing at zero level. In such a situation the given LPP is feasible and an initial b.f.s has been obtained. The second possibility is that it does not happen, i.e. the optimal value of the Phase-I objective function is strictly less than zero or equivalently in the optimal solution of the Phase-I problem some artificial variables x_{a_i} is strictly positive. If this happens then it is an indication that the feasible region of the given LPP is empty or equivalently the constraints of the given LPP are inconsistent. This is because for the standard form LPP, all constraints are equations and so no positive quantity can be added and still maintain the equality sign. For our problem we get the following tableaus for the Phase-I problem

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a_1)}$	$y^{(3)}$	$y^{(a_2)}$	$y^{(4)}$
←	$x_{a_1} = 3$	$\boxed{3}$	1	1	0	0	0
	$x_{a_2} = 6$	4	3	0	−1	1	0
	$x_4 = 3$	1	2	0	0	0	1
	−9	−7	−4	0	1	0	0
		↑					

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a_1)}$	$y^{(3)}$	$y^{(a_2)}$	$y^{(4)}$
	$x_1 = 1$	1	1/3	1/3	0	0	0
←	$x_{a_2} = 2$	0	$\boxed{5/3}$	−4/3	−1	1	0
	$x_4 = 2$	0	5/3	−1/3	0	0	1
	−2	0	−5/3	7/3	1	0	0
			↑				

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a_1)}$	$y^{(3)}$	$y^{(a_2)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	3/5	−1/5	−1/5	0
$x_2 = 6/5$	0	1	−4/5	−3/5	3/5	0
$x_4 = 0$	0	0	1	1	−1	1
0	0	0	1	0	0	0

Here the optimal value of the Phase-I problem is zero and hence the constraints of the original problem are consistent. Further as all the artificial variables are nonbasic, this gives the starting b.f.s for the original problem as $x_1 = 3/5$, $x_2 = 6/5$, $x_4 = 0$. Now we go to Phase-II for finding an optimal solution of the given problem (What happens if instead of x_{a_2}, we take x_4 as the leaving variable? See Example 2.8.1 for the case when some artificial variables are present in the basis at the zero level).

The Phase-II Problem

Here we take the same problem as Phase-I but delete the artificial variables and take the original objective function. The last tableau of Phase-I will serve as the starting tableau for Phase-II except that the last row will be computed again because the objective function has been changed now, and that is the only change as the constraints remain same. Here it may be noted that in the simplex tableau only the entries in last row depend on the objective function; all other entries depend on the constraints only.

For b.f.s $x_1 = 3/5$, $x_2 = 6/5$, $x_4 = 0$ we have $c_B = col(-2, -1, 0)$; $z(x_B) = -12/5$; $z_1 = -2$, $z_2 = 1$, $z_3 = 1/5$, $z_4 = 0$ and therefore the starting tableau for Phase-II is

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	1/5	0
$x_2 = 6/5$	0	1	−3/5	0
$x_4 = 0$	0	0	1	1
−12/5	0	0	1/5	0

Since all $(z_j - c_j) \geq 0$, so the Phase-II problem has been solved. Therefore for the given problem, $(x_1^\star = 3/5, x_2^\star = 6/5)$ is optimal with the optimal value as $z^\star = -(-12/5) = 12/5$.

The Big-M Method

This method is similar to the two phase method except that rather than having two separate problems for Phase-I and Phase-II, here we have a combined problem. After converting the given LPP in the standard form, here again, we introduce appropriate number of artificial variables x_{a_i} in the appropriate constraints so as to get an $(m \times m)$ identity matrix to be taken as an initial basis matrix. But in this approach, the artificial variables are assigned a very large negative cost in the objective function. The simplex method, while trying to improve the objective function, will find the artificial variables uneconomical to maintain as basic variables with the positive value. Hence they will be quickly replaced in the basis by the real (actual) variables with smaller costs. For hand calculations it is not necessary to assign a specific cost value to the artificial variables. The general approach is to take the cost of artificial variables in objective function as $-M$, M being large positive number. However, if the problem is to be solved on a machine then the value of M is to be specified. It is customary to take the value of M as $100 \times \max_j |c_j|$.

Let us solve the below given problem by the Big-M method.

$$\begin{aligned} \text{Max} \quad & z = 3x_1 - x_2 - x_3 \\ \text{subject to} \quad & \\ & x_1 - 2x_2 + x_3 \leq 11 \\ & -4x_1 + x_2 + 2x_3 \geq 3 \\ & -2x_1 + x_3 = 1 \\ & x_1, x_2, x_3 \geq 0. \end{aligned} \tag{2.9}$$

As in the two phase method we first convert the given LPP in the standard form to get

$$\begin{aligned} \text{Max} \quad & z = 3x_1 - x_2 - x_3 + 0x_4 + 0x_5 \\ \text{subject to} \quad & \\ & x_1 - 2x_2 + x_3 + x_4 = 11 \\ & -4x_1 + x_2 + 2x_3 - x_5 = 3 \\ & -2x_1 + x_3 = 1 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0. \end{aligned} \tag{2.10}$$

In the above problem, we note that there is only one identity column given by the slack variable x_4. Therefore we need to introduce two artificial variables x_{a_1} and x_{a_2} in

the second and third constraints respectively. It is obvious that once the LPP is in the standard form then the artificial variables will be introduced in those constraints only which are originally with '≥' or '=' sign, because these will not give identity columns. Only those constraints where the slack variables are introduced will give rise to identity columns. For the problem at hand, we need to solve the following problem by the Big-M method. Here as explained, we take the combined objective function by taking the objective function as given and attaching a very large negative cost to each artificial variable.

$$\text{Max} \quad z = 3x_1 - x_2 - x_3 + 0x_4 + 0x_5 - Mx_{a_1} - Mx_{a_2}$$

subject to

$$\begin{aligned} x_1 - 2x_2 + x_3 + x_4 &= 11 \\ -4x_1 + x_2 + 2x_3 - x_5 + x_{a_1} &= 3 \\ -2x_1 + x_3 + x_{a_2} &= 1 \\ x_1, x_2, x_3, x_4, x_5, x_{a_1}, x_{a_2} &\geq 0. \end{aligned} \tag{2.11}$$

Now taking x_4, x_{a_1}, x_{a_2} as the initial basic variables and solving the problem by the simplex method we obtain the following simplex tableaus

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(a_1)}$	$y^{(a_2)}$
	$x_4 = 11$	1	−2	1	1	0	0	0
	$x_{a_1} = 3$	−4	1	2	0	−1	1	0
←	$x_{a_2} = 1$	−2	0	[1]	0	0	0	1
	$-4M$	$(-3+6M)$	$(1-M)$	$(1-3M)$	0	M	0	0
				↑				

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(a_1)}$	$y^{(a_2)}$
	$x_4 = 10$	3	−2	0	1	0	0	−1
←	$x_{a_1} = 1$	0	[1]	0	0	−1	1	−2
	$x_3 = 1$	−2	0	1	0	0	0	1
	$-1-M$	−1	$(1-M)$	0	0	M	0	$(3M-1)$
			↑					

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(a_1)}$	$y^{(a_2)}$
←	$x_4 = 12$	[3]	0	0	1	−2	2	−5
	$x_2 = 1$	0	1	0	0	−1	1	−2
	$x_3 = 1$	−2	0	1	0	0	0	1
	−2	−1	0	0	0	1	$(M-1)$	$(M+1)$
		↑						

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(a_1)}$	$y^{(a_2)}$
$x_1 = 4$	1	0	0	1/3	−2/3	2/3	−5/3
$x_2 = 1$	0	1	0	0	−1	1	−2
$x_3 = 9$	0	0	1	2/3	−4/3	4/3	−7/3
2	0	0	0	1/3	1/3	$(M - 1/3)$	$(M - 2/3)$

At this stage all $(z_j - c_j) \geq 0$ and so an optimal solution has been obtained. Therefore $x_1^\star = 4$, $x_2^\star = 1$, $x_3^\star = 9$ is an optimal solution and $z^\star = 2$ is the optimal value.

We now take few more examples and solve them by the two phase method. It will be useful if readers solve these by the Big-M method as well.

Example 2.6.1 *Solve the following problem by the simplex method and verify your answer graphically*

$$\begin{aligned} &\textit{Max} && z = 4x_1 + 3x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 \leq 8 \\ & && 2x_1 + x_2 \geq 10 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

Solution It is obvious that we need only one artificial variable x_{a_1} and therefore the Phase-I problem is

$$\begin{aligned} &\text{Max} && z_a = -x_{a_1} \\ &\text{subject to} \\ & && x_1 + x_2 + x_3 = 8 \\ & && 2x_1 + x_2 - x_4 + x_{a_1} = 10 \\ & && x_1, x_2, x_3, x_4, x_{a_1} \geq 0. \end{aligned}$$

Now columns for the variables x_3 and x_{a_1} give identity column and therefore taking these as basic variables we have the following simplex tableaus

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$
	$x_3 = 8$	1	1	1	0	0
←	$x_{a_1} = 10$	2	1	0	−1	1
	−10	−2	−1	0	1	0
		↑				

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$
$x_3 = 3$	0	1/2	1	1/2	−1/2
$x_1 = 5$	1	1/2	0	−1/2	1/2
0	0	0	0	0	1

At this stage all $(z_j - c_j) \geq 0$ and so Phase-I problem has been solved. As the value of the Phase-I objective function is zero, the given LPP is feasible, i.e. the constraints are consistent. Also all artificial variables are nonbasic variables and so we go to Phase-II directly. The Phase-II problem is

$$\begin{aligned} \text{Max} \quad & z = 4x_1 + 3x_2 \\ \text{subject to} \quad & \\ & x_1 + x_2 + x_3 = 8 \\ & 2x_1 + x_2 - x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Here, from the last tableau of Phase-I we drop the artificial columns and reevaluate the elements of the last row with respect to the new c_B (in our example $c_B = (0,4)^T$) to get the initial tableau of the Phase-II problem. This gives the following

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
←	$x_3 = 3$	0	1/2	1	1/2
	$x_1 = 5$	1	1/2	0	−1/2
	20	0	−1	0	−2
					↑

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_4 = 6$	0	1	2	1
$x_1 = 8$	1	1	1	0
32	0	1	4	0

Therefore the optimal solution is $x_1^\star = 8$, $x_2^\star = 0$, and $z^\star = 32$ is the optimal value.

We now solve the given problem graphically. The feasible region is depicted in Fig 2.8 with the corner points $A : (2,6)$, $B : (5,0)$, $C : (8,0)$ respectively. Also $z(A) = 26$, $z(B) = 20$, and $z(C) = 32$. Therefore, the corner point C, namely $x_1^\star = 8, x_2^\star = 0$ is an optimal solution and $z^\star = 32$ is the optimal value. This is the same as obtained by the two phase method.

In Fig. 2.8 we have also shown the corner points corresponding to various simplex tableaus in Phase-I and Phase-II. The initial b.f.s of the Phase-I problem corresponds to the corner point (0,0) which is outside the feasible region. This is because for this b.f.s the artificial variable $x_{a_1} = 10 (> 0)$. But in the next tableau x_{a_1} becomes zero and so we get a feasible corner point (5,0) of the feasible region. After that, in Phase-II, we move from (5,0) to get the optimal corner point (8,0). All this illustrates that as long as some x_{a_i} is a basic variable with positive value, we shall be outside the feasible region. It is only when all $x_{a_1} = 0$, then we occupy a feasible corner point and then do actual optimization in Phase-II.

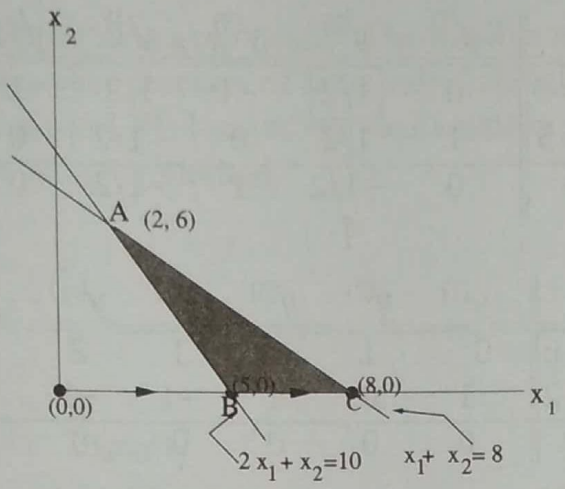


Fig. 2.8.

Example 2.6.2 *Solve the following problem by the simplex method and verify your answer graphically*

$$\begin{aligned} &\textit{Max} && z = 4x_1 + 3x_2 \\ &\textit{subject to} \\ &&& x_1 + x_2 \geq 8 \\ &&& 2x_1 + x_2 \geq 10 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

Solution Here we need two artificial variables x_{a_1} and x_{a_2} and the Phase-I problem is

$$\begin{aligned} &\text{Max} && z_a = -x_{a_1} - x_{a_2} \\ &\text{subject to} \\ &&& x_1 + x_2 - x_3 + x_{a_1} = 8 \\ &&& 2x_1 + x_2 - x_4 + x_{a_2} = 10 \\ &&& x_1, x_2, x_3, x_4, x_{a_1}, x_{a_2} \geq 0. \end{aligned}$$

Now in the initial b.f.s x_{a_1} and x_{a_2} are the basic variables. This gives the following simplex tableaus

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$	$y^{(a_2)}$
	$x_{a_1} = 8$	1	1	−1	0	1	0
←	$x_{a_2} = 10$	**2**	1	0	−1	0	1
	−18	−3	−2	1	1	0	0
		↑					

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$	$y^{(a_2)}$
←	$x_{a_1} = 3$	0	$\boxed{1/2}$	−1	1/2	1	−1/2
	$x_1 = 5$	1	1/2	0	−1/2	0	1/2
	−3	0	−1/2	1	−1/2	0	3/2
			↑				

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$	$y^{(a_2)}$
$x_2 = 6$	0	1	−2	1	2	−1
$x_1 = 2$	1	0	1	−1	−1	1/2
0	0	0	0	0	0	0

As the optimal objective function value of the Phase-I problem is zero, the given LPP is feasible. Also both artificial variables are nonbasic variables and so we drop them and go to Phase-II as explained. The Phase-II problem is

$$\text{Max} \qquad z = 4x_1 + 3x_2 + 0x_3 + 0x_4$$

subject to

$$x_1 + x_2 - x_3 = 8$$
$$2x_1 + x_2 - x_4 = 10$$
$$x_1, x_2, x_3, x_4 \geq 0.$$

Now, the initial tableau of the Phase-II problem is

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_2 = 6$	0	1	−2	1
←	$x_1 = 2$	1	0	$\boxed{1}$	−1
	26	0	0	−2	−1
				↑	

Here the last row has been written w.r.t the objective function of the Phase-II problem by taking $c_B = (3,4)^T$.

After pivoting we get the next tableau as

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 10$	2	1	0	−1
$x_3 = 2$	1	0	1	−1
30	2	0	0	−3

Now for the last column, namely $y^{(4)}$, Result 2.5.2 becomes applicable because $z_4 - c_4 = -3 < 0$ and in this column no y_{ij} is more than zero. As a consequence, using Result 2.5.2, we conclude that the given LPP has unbounded solution.

We now solve the above problem graphically. The feasible region is as depicted in Fig 2.9 and the arrow indicates the direction of increase of the given objective function. It clearly indicates that the given LPP has unbounded solution which matches with what has been indicated by the simplex method.

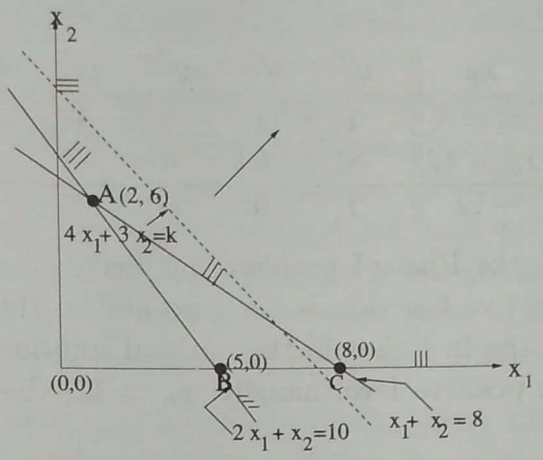


Fig. 2.9.

Example 2.6.3 *Solve the following problem by the simplex method and verify your answer graphically*

$$\begin{aligned} &\textit{Max} && z = 4x_1 + 3x_2 \\ &\textit{subject to} \\ &&& x_1 + x_2 \leq 8 \\ &&& 5x_1 + 6x_2 \geq 60 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

Solution Here we need only one artificial variable x_{a_1} and therefore the Phase-I problem is

$$\begin{aligned} &\text{Max} && z_a = -x_{a_1} \\ &\text{subject to} \\ &&& x_1 + x_2 + x_3 = 8 \\ &&& 5x_1 + 6x_2 - x_4 + x_{a_1} = 60 \\ &&& x_1, x_2, x_3, x_4, \quad x_{a_1} \geq 0. \end{aligned}$$

Taking x_3 and x_{a_1} as initial basic variables we have the following simplex tableaus for the Phase-I problem.

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$
←	$x_3 = 8$	1	1	1	0	0
	$x_{a_1} = 60$	5	6	0	−1	1
	−60	−5	−6	0	1	0
			↑			

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a_1)}$
$x_2 = 8$	1	1	1	0	0
$x_{a_1} = 12$	−1	0	−6	−1	1
−12	1	0	6	1	0

Now, as all $(z_j - c_j) \geq 0$, the Phase-I problem has been solved. But here the optimal value of the Phase-I objective function is not zero and so the given LPP is infeasible. This is also indicated by the fact that in the optimal solution of the Phase-I problem, x_{a_1} is still present at the positive level namely, $x_{a_1} = 12$, thereby making the original constraint inconsistent.

We now solve the given problem graphically. The feasible region is depicted in Fig. 2.10. As the two lines $x_1 + x_2 = 8$, $5x_1 + 6x_2 = 60$ do not intersect in the first quadrant, there is no point in $\mathbf{R}^2$ satisfying all the constraints and so the set of feasible solutions is an empty set. Thus the given LPP is infeasible or equivalently the constraints are inconsistent.

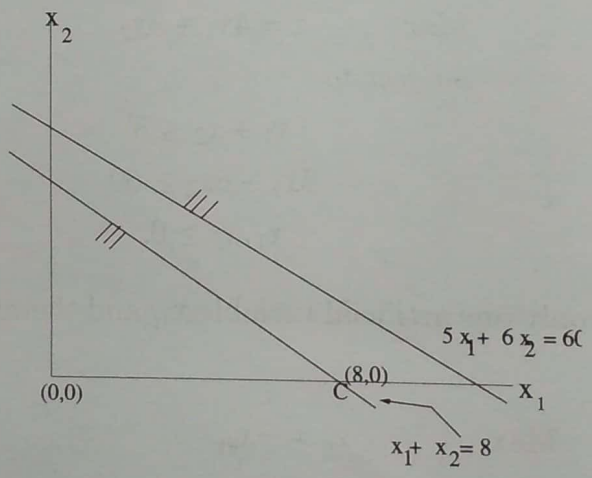


Fig. 2.10.

2.7 Alternate Optima

While discussing the graphical method in Section 2.2, we noted that if the given LPP has an optimal solution then either it has unique optimal solution or it has infinitely many

optimal solutions. In the second scenario, i.e. when there are infinitely many optimal solutions, we say that the given LPP has *alternative optima*. Now a glance at Fig. 2.5 suggests that not all alternative optimal solutions are corner points (b.f.s). In fact in this figure, the set of all optimal solutions is the line segment joining the two optimal corner points. In general, it can be shown that set of all optimal solutions is the convex set spanned by the optimal corner points, i.e. it is the *convex hull* of the optimal corner points.

If the given LPP is such that it can be solved by the graphical method, then certainly all optimal corner points can be identified geometrically and hence the relevant convex hull, i.e. the set of all optimal solutions can also be viewed geometrically. The obvious question here is to know if the simplex method can also do the same. The answer is 'yes'. From the optimal simplex tableau, using the below given result, we can infer if the given LPP has alternative optima or not. Further, if there is indication that alternative optima do exist then we can also determine all optimal basic feasible solutions.

Result 2.7.1 *In the optimal simplex tableau of the given LPP, if for some non-basic variable* x_j, $z_j - c_j = 0$ *and for that some* $y_{ij} > 0$ *then the problem has alternative optima.*

We now take following example to explain some of the points discussed above.

Example 2.7.1 *Use the simplex method to solve the following problem and verify your results graphically*

$$\begin{aligned} &Max && z = x_1 + x_2 \\ &\text{subject to} \\ &&& x_1 + x_2 \leq 8 \\ &&& 2x_1 + x_2 \leq 10 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

Solution Here we need two slack variables to get the following standard form LPP

$$\begin{aligned} &\text{Max} && z = x_1 + x_2 + 0x_3 + 0x_4 \\ &\text{subject to} \\ &&& x_1 + x_2 + x_3 = 8 \\ &&& 2x_1 + x_2 + x_4 = 10 \\ &&& x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Now in the initial b.f.s, x_3 and x_4 are basic variables. This gives the following simplex tableaus

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_3 = 8$	1	1	1	0
$\leftarrow$	$x_4 = 10$	$\boxed{2}$	1	0	1
	0	−1	−1	0	0
		$\uparrow$			

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
←	$x_3 = 3$	0	$\boxed{1/2}$	1	$-1/2$
	$x_1 = 5$	1	1/2	0	1/2
	5	0	$-1/2$	0	1/2
			↑		

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_2 = 6$	0	1	2	-1
←	$x_1 = 2$	1	0	-1	$\boxed{1}$
	8	0	0	1	0
					↑

So an optimal solution of the given LPP is $(x_1^\star = 2, x_2^\star = 6)$ and the optimal value is $z^\star = 8$. Also Result 2.7.1 is applicable indicating that the given problem has alternative optima. Since the hypothesis of Result 2.7.1 holds for the nonbasic variable x_4, there exists another optimal b.f.s in which x_4 is a basic variable. To find this optimal b.f.s we make x_4 a basic variable in the next iteration and as per the minimum ratio criteria make x_1 a nonbasic variable to get the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 8$	1	1	1	0
$x_4 = 2$	1	0	-1	1
8	0	0	1	0

Thus we obtain another optimal b.f.s given by $(x_1^\star = 0, x_2^\star = 8)$.

If we now solve the given problem by graphical method, we obtain P: (2,6) and Q: (0, 8) as two optimal corner points, which correspond to two basic feasible solutions obtained by the simplex method. Also as we see in Fig 2.11, all points on the line segment joining P and Q are also optimal. Infact, as we shall prove later, the set of all optimal solutions of a LPP is a convex set and therefore if the given LPP has more than one optimal solution then it has infinitely many of them. In our example, P and Q are optimal corner points and all the points on the line segment joining P and Q are optimal. Algebraically a point on this line segment (except P and Q) will correspond to a feasible optimal solution and NOT a basic feasible optimal solution.

2.8 Redundancy in Linear Programming

While applying the simplex method, we have emphasized that the given LPP must be in the standard form, i.e. it has the form

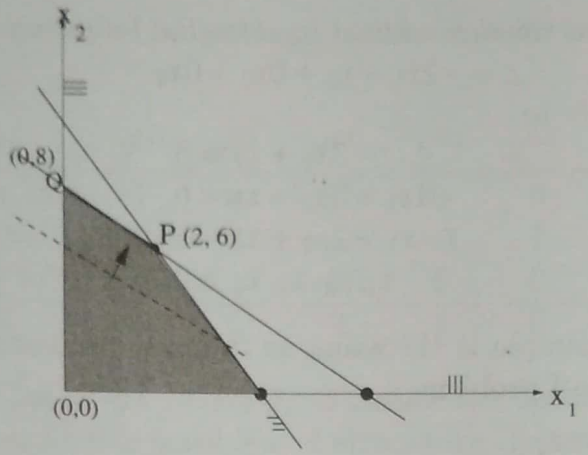


Fig. 2.11.

$$\begin{aligned} \text{Max} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.12}$$

with (i) $b \geq 0$ and (ii) Rank $A = m(< n)$. Sometimes, it may happen that Rank A may be less than m, i.e. rows of A are linearly dependent or equivalently there is redundancy in the constraints of (2.12). In this section we wish to understand this scenario and show that the simplex method itself can be used to detect the same.

Let us now go back to the two phase method discussed in Section 2.6 and note that if the given LPP is feasible then at the end of Phase I, the optimal value of the Phase-I objective function is certainly zero. This implies that in the optimal solution of the Phase-I problem all artificial variables x_{a_i} take the zero value. But a variable x_{a_i} may take the zero value in two ways. Either it is a nonbasic variable (hence it is zero) or it is a basic variable at the zero level. If in the optimal Phase-I tableau all x_{a_i} take the zero value as nonbasic variables then we drop the corresponding columns from the last Phase-I tableau and go to Phase-II directly as explained in Section 2.6.

However sometime it may happen that although the optimal value of the Phase-I objective function is zero but some artificial variable is present in the basis at the zero level. In such a situation we can not delete the corresponding artificial column for going to Phase-II, because then we shall be short of one basic variable. So what should we do in these situations. It is obvious that we can not go to Phase-II unless we have a genuine starting b.f.s, i.e. unless we 'exchange' all such x_{a_i} (i.e. those x_{a_i} which are basic variables at zero level) by genuine variables. If 'exchange' is possible then we first do the 'exchange' and then go to Phase-II. If in some problem 'exchange' is not possible then it is an indication of redundancy in the problem. We explain the 'exchange' process and redundancy with the help of following examples.

Example 2.8.1 *Use the simplex method to solve the following*

$$\begin{aligned} &\textit{Max} && z = -2x_1 - x_2 + 0x_3 + 0x_4 \\ &\textit{subject to} \\ & && 3x_1 + x_2 = 3 \\ & && 4x_1 + 3x_2 - x_3 = 6 \\ & && x_1 + 2x_2 + x_4 = 3 \\ & && x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Solution The above example is the same as the one discussed in Section 2.6. So we have the following Phase-I problem

$$\begin{aligned} &\text{Max} && z_a = -x_{a1} - x_{a2} \\ &\text{subject to} \\ & && 3x_1 + x_2 + x_{a1} = 3 \\ & && 4x_1 + 3x_2 - x_3 + x_{a2} = 6 \\ & && x_1 + 2x_2 + x_4 = 3 \\ & && x_1, x_2, x_3, x_4, x_{a1}, x_{a2} \geq 0 \end{aligned}$$

and the corresponding simplex tableaus are

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a1)}$	$y^{(3)}$	$y^{(a2)}$	$y^{(4)}$
←	$x_{a1} = 3$	[3]	1	1	0	0	0
	$x_{a2} = 6$	4	3	0	−1	1	0
	$x_4 = 3$	1	2	0	0	0	1
	−9	−7	−4	0	1	0	0
		↑					

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a1)}$	$y^{(3)}$	$y^{(a2)}$	$y^{(4)}$
	$x_1 = 1$	1	1/3	1/3	0	0	0
	$x_{a2} = 2$	0	5/3	−4/3	−1	1	0
←	$x_4 = 2$	0	[5/3]	−1/3	0	0	1
	−2	0	−5/3	7/3	1	0	0
			↑				

Now x_2 is the entering variable as this corresponds to the negative most value of $(z_j - c_j)$. But which variable should be taken out as the minimum ratio, i.e. $\min(2/(5/3), 2/(5/3))$ is same for variable x_{a2} as well as for the variable x_4. Theoretically we can make any of these two variables, namely, x_{a2} and x_4, as nonbasic variables in the next tableau. Earlier in Section 2.6, we chose x_{a2} as the leaving variables (it makes sense as we wish to make x_{ai} out of the basis as early as possible) and got the starting b.f.s as $(x_1 = 3/5, x_2 = 6/5, x_4 = 0)$.

But then we can choose x_4 to be leaving variable as well. Let us now do that so as to get the tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a1)}$	$y^{(3)}$	$y^{(a2)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	2/5	0	0	−1/5
$x_{a2} = 0$	0	0	−1	−1	1	−1
$x_2 = 6/5$	0	1	−1/5	0	0	3/5
0	0	0	2	1	0	1

Now Phase-I is over (as all $(z_j - c_j) \geq 0$) and since the optimal value of the Phase-I problem is zero, the given LPP is feasible and all x_{ai} take the zero value. But then there is some difference between this tableau and the tableau obtained in Section 2.6.

Here, though all x_{ai}, i.e. x_{a1} and x_{a2} are zero, the variable x_{a2} is zero as a basic variable so we can not just drop x_{a2} and the column $y^{(a2)}$ and go to Phase-II because then we shall be short of one basic variable.

In such situation we verify if the 'exchange' is possible. If 'exchange' is possible, then we first do the 'exchange' and then go to Phase-II. If in some problem 'exchange' is not possible then that is the indication of redundancy.

In 'exchange' we exchange a variable like x_{a2} (i.e. an artificial variable which is present in last Phase-I tableau at the zero level, but as a basic variable) by doing one more iteration of the simplex algorithm so that the optimal value (which is zero) of the Phase-I objective function is not disturbed. For this, we make x_{a2} as a leaving variable (i.e. make it a nonbasic variable in the next iteration) and choose any current nonbasic variable (e.g. x_3 or x_4 but NOT x_{a1}) as entering variable (i.e. make it a basic variable in the next iteration). This is possible only when in the row of x_{a2}, the corresponding y_{ij} for x_3 or x_4 is not zero. If no such nonzero y_{ij} exist for any of the genuine nonbasic variables (other than those artificial variables which are nonbasic), then we say that exchange is not possible and we declare redundancy.

In our example we can exchange x_{a2} either with x_3 or x_4. Suppose we decide to exchange x_{a2} with x_4. This gives the following new tableau for the Phase-I problem

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(a1)}$	$y^{(3)}$	$y^{(a2)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	3/5	1/5	−1/5	0
$x_4 = 0$	0	1	1	1	−1	1
$x_2 = 6/5$	0	1	−4/5	−3/5	3/5	0
0	0	0	1	0	1	0

Now all artificial variable are zero as nonbasic variables and so we can drop the corresponding artificial columns from the above tableau and go to Phase-II as discussed earlier. This gives the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	1/5	0
$x_4 = 0$	0	1	1	1
$x_2 = 6/5$	0	1	−3/5	0
−12/5	0	0	1/5	0

Therefore, $(x_1^* = 3/2, x_2^* = 6/5)$ is an optimal solution of the given LPP with the optima value $z^* = -12/5$.

Example 2.8.2 *Use the simplex method to solve the following*

$$\begin{aligned} &\textit{Max} && z = x_1 - 2x_2 + 3x_3 \\ &\textit{subject to} \\ & && x_1 + x_2 + x_3 = 6 \\ & && -x_1 + x_2 + 2x_3 = 4 \\ & && 2x_2 + 3x_3 = 10 \\ & && x_1, x_2, x_3 \geq 0. \end{aligned}$$

Solution We first write the given LPP in the standard form, i.e.

$$\begin{aligned} &\text{Max} && z = x_1 - 2x_2 + 3x_3 + 0x_4 \\ &\text{subject to} \\ & && x_1 + x_2 + x_3 = 6 \\ & && -x_1 + x_2 + 2x_3 = 4 \\ & && 2x_2 + 3x_3 = 10 \\ & && x_3 + x_4 = 2 \\ & && x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Here $A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ -1 & 1 & 2 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ and Rank A=3 where as the number of constraints, i.e. m equals 4. So there is redundancy we can also verify the same by noting that the sum of the first two rows is the same as the third row. In the following we use the simplex method to identify redundancy. For this we construct the Phase-I problem

$$\begin{aligned} &\text{Max} && z_a = -x_{a1} - x_{a2} - x_{a3} \\ &\text{subject to} \\ & && x_1 + x_2 + x_3 + x_{a1} = 6 \\ & && -x_1 + x_2 + 2x_3 + x_{a2} = 4 \\ & && 2x_2 + 3x_3 + x_{a3} = 10 \\ & && x_3 + x_4 = 2 \\ & && x_1, x_2, x_3, x_4, x_{a1}, x_{a2} \geq 0 \end{aligned}$$

and get the following last tableau for the same as

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(a1)}$	$y^{(a2)}$	$y^{(a3)}$
$x_1 = 2$	1	0	0	1/2	1/2	−1/2	0
$x_2 = 2$	0	1	0	−3/2	1/2	1/2	0
$x_{a3} = 0$	0	0	0	0	−1	−1	1
$x_3 = 2$	0	0	1	1	0	0	0
0	0	0	0	0	2	2	0

As at the end of Phase-I, the objective function value is zero, the given LPP is feasible. But before going to Phase-II, we have to exchange x_{a3} as it is zero as a basic variable. For this in the row for the variable x_{a3} in the tableau, we have to check if the y_{ij} value is nonzero for some genuine (i.e. not an artificial variable) nonbasic variable. If no such value exists, then there is redundancy in the LPP. In the above tableau, the only variable which can be exchanged with x_{a3} is x_4 but for that the corresponding y_{ij} value is zero. So here exchange is NOT possible and so the given LPP has redundancy.

Since x_{a2} is the artificial variable which is zero as a basic variable and that can not be exchanged, the original constraint where x_{a2} has been added, is redundant. Therefore to solve the given LPP we should first drop that redundant constraint and then solve the problem. Equivalently, we drop the third row (row for x_{a2}) from the above tableau and then go to Phase-II to get the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_1 = 2$	1	0	0	1/2
$x_2 = 2$	0	1	0	3/2
$x_3 = 2$	0	0	1	1
4	0	0	0	13/2

Therefore, $(x_1^\star = 2, x_2^\star = 2, x_3^\star = 2)$ is an optimal solution of the given LPP with the optimal value $z^\star = 4$.

2.9 Degeneracy and Cycling

In the context of the equivalence between basic feasible solutions and corner points we have observed that two non-degenerate b.f.s. will correspond to two distinct corner points but more than one degenerate b.f.s. *may* correspond to the same corner point. Therefore, in the absence of degeneracy, the simplex method will terminate in finite number of iterations because then there will be strict improvement in the value of objective function for the successive b.f.s. (i.e. successive corner points which will be distinct) and the number of b.f.s. is always finite.

But there may be some difficulty in the presence of degeneracy. To understand this situation, let us assume that the current b.f.s. x_B is degenerate and the current objective

function value is $z(x_B)$. Let us also assume that for the negative most value of $(z_j - c_j)$ (say $(z_k - c_k)$) , $y_{rk} > 0$ and $x_{B_r} = 0$ (there is certainly a possibility of such a scenario). Therefore as per the minimum ratio criteria the variable x_{B_r} will become non-basic in the next tableau. Also the variable x_k will become a basic variable with the pivot element as $y_{rk}(> 0)$. Then using the pivoting rule we get $\widehat{x_{B_r}} = x_k = 0$ and $z(\widehat{x_B}) = z(x_B)$. This shows that the new solution $\widehat{x_B}$ is also degenerate and there is no change in the objective function value. Infact both degenerate b.f.s x_B and $\widehat{x_B}$ correspond to the same corner point. So though we have performed one iteration algebraically we have not left the current corner point at all.

Now what we have assumed for x_B can certainly happen for $\widehat{x_B}$ so that the new b.f.s $\widehat{\widehat{x_B}}$ is also degenerate and all three b.f.s, namely, x_B, $\widehat{x_B}$, and $\widehat{\widehat{x_B}}$ correspond to the same corner point. Continuing with this argument we infer that there is certainly a possibility (though very rare) that we get involve into a sequence of degenerate b.f.s $\{x_B^{(1)}, x_B^{(2)}, \ldots, x_B^{(k-1)}, x_B^{(k)}\}$ all corresponding to the same corner point, and $x_B^{(k)} = x_B^{(1)}$. In this situation we shall repeat the same sequence and continue indefinitely. This phenomenon is called *cycling* in the simplex method.

If for some problem we are trapped in cycling, then the simplex method will never terminate because though algebraic iterations are being performed, geometrically, there is no movement as all degenerate b.f.s. in the sequence correspond to the same corner point.

Thus except for cycling, the simplex method will always terminate in finite number of iterations. Here the readers should appreciate two points. Firstly cycling occurs due to degeneracy but not everytime degeneracy will lead to cycling. We need that rare coincidence when all degenerate b.f.s. meet the stated conditions so that all correspond to the same corner point. Second point to appreciate here is that degeneracy is very common in applying the simplex method to real life problems but cycling is very-very uncommon. Nevertheless since cycling may occur in the simplex method, ways have been devised to augment the simplex method so that cycling is avoided, e.g. Hadley [72] but we do not plan to discuss these here.

We now present an artificially constructed example to illustrate cycling in the simplex method. For this let us consider the LPP

$$\text{Min} \quad z = 3/4x_1 - 20x_2 + 1/2x_3 - 6x_4$$

subject to

$$\begin{aligned} 1/2x_1 - 12x_2 - 1/2x_3 + 3x_4 &\leq 0 \\ 1/4x_1 - 8x_2 - x_3 + 9x_4 &\leq 0 \\ x_3 &\leq 1 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Introducing the slack variables x_5, x_6, and x_7 and writing the problem in the standard form, we get

$$\text{Min} \qquad z = 3/4x_1 - 20x_2 + 1/2x_3 - 6x_4$$

subject to

$$1/2x_1 - 12x_2 - 1/2x_3 + 3x_4 + x_5 = 0$$
$$1/4x_1 - 8x_2 - x_3 + 9x_4 + x_6 = 0$$
$$x_3 + x_7 = 1$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0.$$

If we solve the above problem by the simplex method then the results of various tableaus can be summarized as follows

Iteration no	x_B	$z(x_B)$	Corner point
1	$x_5 = 0, x_6 = 0, x_7 = 1$	0	(0,0,0,0)
2	$x_5 = 0, x_1 = 0, x_7 = 1$	0	(0,0,0,0)
3	$x_2 = 0, x_1 = 0, x_7 = 1$	0	(0,0,0,0)
4	$x_2 = 0, x_3 = 0, x_7 = 1$	0	(0,0,0,0)
5	$x_4 = 0, x_3 = 0, x_7 = 1$	0	(0,0,0,0)
6	$x_4 = 0, x_6 = 0, x_7 = 1$	0	(0,0,0,0)
7	$x_5 = 0, x_6 = 0, x_7 = 1$	0	(0,0,0,0)

Here the b.f.s obtained at the seventh iteration is the same as obtained at the initial iteration (i.e. iteration number 1) and so we have a sequence of six degenerate b.f.s which are going to cycle and they all correspond to the same corner point (0,0,0,0) in $\mathbf{R}^4$ where the objective function value is zero.

2.10 The Simplex Tableau in the Condensed Form

Let us recall the format of the simplex tableau used in earlier sections and call it as *tableau in the extended form.* In these tableaus we store $y^{(j)}$ columns for both basic as well as nonbasic variables. But it is already known that if $x^{(j)}$ is a basic variable then $y^{(j)}$ column is an identity column and the value of $(z_j - c_j) = 0$. Therefore storing $y^{(j)}$ columns and the values of $(z_j - c_j)$ for basic variables x_j is rather unnecessary and can possibly be avoided. Keeping this in mind, in this section we introduce a new format for the simplex tableau and that we shall call as the *tableau in the condensed form.*

The main advantage of using the tableau in condensed form is that we have to store lesser data and work with a smaller sized tableau. The only change in the earlier working will be with regard to pivoting. The pivoting rules used earlier for the tableau in extended form will need modification if we are working with the tableau in the condensed form.

We take following example to illustrate the format of the tableau in the condensed form and also various steps for pivoting in this format.

$$\begin{aligned} \text{Max} \quad & z = 4x_1 + 3x_2 + 0x_3 + 0x_4 \\ \text{subject to} \quad & \\ & x_1 + x_2 + x_3 = 8 \\ & 2x_1 + x_2 + x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

If we are working with the usual tableau (i.e. tableau in the extended form) then the initial tableau will look like

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 8$	1	1	1	0
$x_4 = 10$	2	1	0	1
0	−4	−3	0	0

If we now decide to work with the tableau in the condensed form then the initial tableau will look like

x_B	$y^{(1)}$	$y^{(2)}$
$x_3 = 8$	1	1
$x_4 = 10$	2	1
0	−4	−3

which we shall normally write as

		1	2	
	8	1	1	3
←	10	**2**	1	4
	0	−4	−3	
		↑		

Here our understanding is that the indices in the extreme right column identify the current basic variables and the indices in the topmost row identify the current nonbasic variables. Thus with respect to the above tableau, we read it as 'x_3 is a basic variable whose value is 8 and x_4 is a basic variable whose value is 10 and x_1 and x_2 are nonbasic variables'. These indices (in the right most column and topmost row) help us in keeping track of basic and nonbasic variables at every iteration, because while pivoting we also exchange the indices of leaving and entering variables.

Now we identify the column to enter and column to leave, and hence the pivot element, in the usual manner. In our example, x_1 is going to be a basic variable and x_4 is going to be a nonbasic variable in the next tableau. These are indicated by 'arrows' as before. Here the pivot element is 2 as indicated.

The modified rules of pivoting for tableau in the condensed form are as under

Step 1 Replace the pivot element by its reciprocal.

Step 2 Divide the remaining entries of the pivot row by the pivot element.

Step 3 Divide the remaining entries of the pivot column by the negative of the pivot element.

Step 4 For remaining entries of the tableau, follow the same update rule as the one for tableau in the extended form.

Step 5 Exchange the indices of the variable to enter and variable to leave.

Here we may note that except for Step 5, nothing is done (in Steps 2,3 or 4) for indices in the right most column and top most row. As such these are not entries in the tableau, they just identify the basic and nonbasic variables at every iteration.

For our example, after performing the above pivoting steps on the initial tableau in the condensed form we get the following

	4	2	
← 3	−1/2	1/2	3
5	1/2	1/2	1
20	2	−1	
		↑	

In the tableau x_3 is a basic variable whose value is 3 and x_1 is a basic variable whose value is 5. Also x_4 and x_2 are now basic variables. As $(z_2 - c_2)$ is still negative, the current solution is not optimal. As indicated, now x_2 becomes a basic variable and x_3 becomes a nonbasic variable, and therefore the modified pivoting rules as described here give the next tableau as

	4	3	
6	−1	2	2
2	1	−1	1
20	1	2	

Therefore $x_1^\star = 2$, $x_2^\star = 6$ is an optimal solution and the optimal value is $z^\star = 20$.

2.11 Summary and Additional Notes

- Section 2.1 presents the graphical method which provides certain intuitive results for the general LPP's.
- Section 2.3 describes the simplex method as an algebraic version of the graphical method. The concept of b.f.s is introduced as an algebraic analogue of the geometric concept 'corner point'.
- Sections 2.4-2.7 present the complete working of the simplex method, while Section 2.8 and Section 2.9 respectively discuss the redundancy and the cycling in linear programming.
- Section 2.10 explains the condensed form of the tableau which is seemingly more useful for writing code of the simplex algorithm.

- The simplex algorithm was developed by G.B. Dantzig in 1947 and published at a later date in 1949.
- There are some excellent texts on linear programming, e.g., Hadley [72], Murty [117], Bazaraa et al. [12], and Gass [66].
- The most classic book on linear programming is by Dantzig himself, i.e. Dantzig [44] which is an experience to read.
- Linear programming is also covered in most of the text books on operations research. In particular Taha [154] and Phillips et al.[122] are excellent references.
- The book by Charnes and Cooper [35] gives an excellent account of various applications of linear programming in management and industry.

2.12 Exercises

2.1 *Solve the following LPP's graphically*

(1) *Max* $z = 2x_1 + 4x_2$
subject to
$$3x_1 + 5x_2 \leq 15$$
$$3x_1 + 2x_2 \leq 12$$
$$x_1, x_2 \geq 0.$$

(2) *Min* $z = x_1 - 10x_2$
subject to
$$x_1 - 5x_2 \geq 0$$
$$-x_1 + 5x_2 \leq 5$$
$$x_1, x_2 \geq 0.$$

(3) *Max* $z = 2x_1 + 5x_2$
subject to
$$x_1 + 2x_2 \leq 20$$
$$x_1 + x_2 \leq 15$$
$$x_2 \leq 6$$
$$x_1, x_2 \geq 0.$$

(4) *Max* $z = x_1 + 2x_2$
subject to
$$x_1 + x_2 + x_3 = 8$$
$$2x_1 + x_2 + x_4 = 10$$
$$x_1, x_2, x_3, x_4 \geq 0.$$

(5) *Max* $z = 2x_1 + x_2$
subject to
$$2x_1 + x_2 \leq 4$$
$$-x_1 + x_2 \geq 1$$
$$x_1, x_2 \geq 0.$$

(6) *Max* $z = 2x_1 + x_2$
subject to
$$x_1 + x_2 \geq 1$$
$$-x_1 + x_2 \geq 1$$
$$x_1, x_2 \geq 0.$$

(7) *Max* $z = 6x_1 - 2x_2$
subject to
$$x_1 - x_2 \leq 1$$
$$2x_1 - x_2 \leq 6$$
$$x_1, x_2 \geq 0.$$

(8) *Max* $z = 6x_1 - 2x_2$
subject to
$$2x_1 - x_2 \leq 2$$
$$x_1 \leq 4$$
$$x_1, x_2 \geq 0.$$

(9) *Max* $z = x_1 - 2x_2$
subject to
$$x_1 + x_2 \geq 2$$
$$-x_1 + x_2 \geq 1$$
$$x_2 \leq 3$$
$$x_1, x_2 \geq 0.$$

(10) *Max* $z = x_2$
subject to
$$|x_1| + |x_2| \leq 2$$
$$x_1 \geq 0.$$

2.2 *Let the function* $f(x_1, x_2) = (-3x_1 + x_2)$ *be minimized over the solid triangular region* ABC *with* $A : (-1, 0)$, $B : (2, 0)$, $C : (0, 1)$. *Formulate this as a LPP and solve the same graphically.*

2.3 *Consider the problem*

$$\begin{aligned} &\textit{Max} && z = Min(3x - 10, -5x + 5) \\ &\textit{subject to} \\ & && 0 \le x \le 5. \end{aligned}$$

(a) Solve the above problem graphically.
(b) Formulate the above as a LPP in standard form.

2.4 *Suppose we want to show that all solutions of*

$$\begin{aligned} x_1 + x_2 &\le 4 \\ 2x_1 - 3x_2 &\le 6 \\ x_1 \ge 0, x_2 &\ge 0 \end{aligned}$$

also satisfy $x_1 + 2x_2 \le 8$. *Formulate this problem as a LPP and verify your result graphically.*

2.5 *Use the simplex method to solve all problems given in 2.1 and identify those which have*

1. *unique optimal solution*
2. *unbounded solution*
3. *infinitely many optimal solutions and*
4. *no feasible solution.*

Verify your answers, both by the simplex method as well as by the graphical method.

2.6 *Identify all basic feasible solutions for the system*

$$\begin{aligned} x_1 + 4x_2 + x_3 &= 8 \\ x_1 + 2x_2 + x_4 &= 4 \\ x_1, x_2, x_3, x_4 &\ge 0 . \end{aligned}$$

2.7 *Solve the following LPP without using the simplex method*

$$\begin{aligned} &\textit{Max} && z = 4x_1 + 5x_2 + 11x_3 + 2x_4 \\ &\textit{subject to} \\ & && 21x_1 + 7x_2 - 3x_3 + 10x_4 = 210 \\ & && x_i \ge 0, i = 1, 2, 3, 4 . \end{aligned}$$

2.8 *Are the following constraints consistent? Use the simplex method to check the same.*

$$2x_1 - 3x_2 \ge 2$$

$$-x_1 + x_2 \ge 3$$

$$x_1 \ge 0, x_2 \ge 0.$$

2.9 *Use the simplex method to check the consistency of the following system*

$$\begin{aligned} -6x_1 + x_2 + x_3 &\leq 5 \\ -2x_1 + 2x_2 - 3x_3 &\geq 3 \\ 2x_2 - 4x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

2.10 *Solve the following LPP*

$$\text{Min} \quad z = 2x_1 - x_2 + 2x_3$$

subject to

$$\begin{aligned} -x_1 + x_2 + x_3 &= 4 \\ -x_1 + x_2 - x_3 &\leq 6 \\ x_1 \leq 0, x_2 &\geq 0 \end{aligned}$$

x_3 *unrestricted in sign.*

2.11 *Use the simplex method to determine a solution of the following set of linear equations*

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + x_2 &= 10\,. \end{aligned}$$

2.12 *Find all degenerate b.f.s of the system*

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ x_1 - x_2 + x_4 &= 0 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

2.13 *Solve the following LPP by the two phase method and illustrate each iteration graphically*

$$\text{Max} \quad z = -x_1 + 8x_2$$

subject to

$$\begin{aligned} x_1 + x_2 &\geq 1 \\ -x_1 + 6x_2 &\leq 3 \\ x_1 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

2.14 *Write the solution of following LPP without actually solving it*

$$\text{Max} \quad z = x_1 - x_2 + x_3 - x_4$$

subject to

$$\begin{aligned} 0 \leq x_1 &\leq 8 \\ -2 \leq x_2 &\leq 4 \\ -2 \leq x_3 &\leq 4 \\ 0 \leq x_4 &\leq 10\,. \end{aligned}$$

2.15 *Is the following a LPP ?*

$$\begin{aligned} \textit{Max} \quad & z = 4x_1 + 3x_2 \\ \textit{subject to} \quad & (x_1 + x_2 \leq 8 \quad \textit{or} \quad 2x_1 + x_2 \leq 10) \\ & x_1, x_2 \geq 0 \,. \end{aligned}$$

Solve the given problem graphically.

2.16 *Solve the LPP*

$$\begin{aligned} \textit{Max} \quad & z = x_1 + 2x_2 + \ldots\ldots + nx_n \\ \textit{subject to} \quad & \\ & x_1 + x_2 + \ldots\ldots + x_n \leq 1 \\ & x_2 + \ldots\ldots + x_n \leq 2 \\ & \ldots \\ & x_n \leq n \\ & x_i \geq 0,\ i = 1, 2\ldots\ldots, n. \end{aligned}$$

3
Mathematics of the Simplex Method

3.1 Introduction

While describing the simplex method in the last chapter, we have used many results which were guessed purely from the geometry of the linear programming problem. The basic aim of this chapter is to prove all these results mathematically so as to complete the discussion of the simplex method from theoretical point of view as well.

3.2 Some Basic Definitions

In this section we introduce some basic definitions on convex sets and related concepts, which are to be used in the subsequent sections.

Definition 3.2.1 (Convex Set). *Let $S \subseteq \mathbf{R}^n$. The set S is called a convex set if for $0 \leq \lambda \leq 1, x, u \in S \Rightarrow \lambda x + (1-\lambda)u \in S$.*

Thus a set $S \subseteq \mathbf{R}^n$ is convex if for any two points x, u in S, the whole line segment joining x and u is in the set S. In Fig 3.1, the first and third sets are convex in $\mathbf{R}^2$ but the second set (the shaded portion) is not convex.

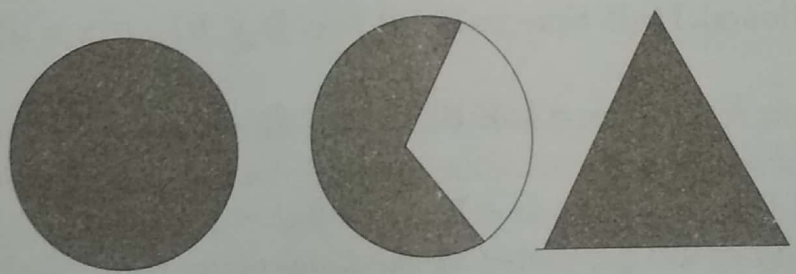


Fig. 3.1.

By convention, an empty set and a single point set are always considered to be convex. Also the intersection of arbitrary many convex sets is always a convex set but the union may not be so.

Definition 3.2.2 (Convex Hull). *Let $S \subseteq \mathbf{R}^n$. Then the smallest convex set containing the given set S is called the convex hull of S and is denoted by $Conv(S)$.*

It is obvious that if S is a convex set then $\text{Conv}(S) = S$. In Fig 3.2 a nonconvex set and its convex hull are depicted in $\mathbf{R}^2$.

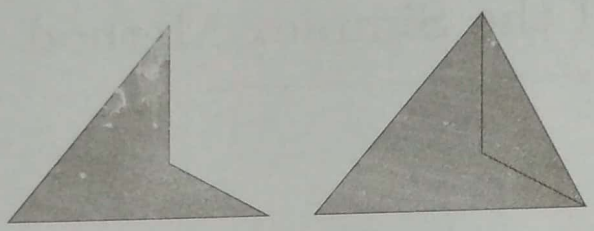


Fig. 3.2.

Definition 3.2.3 (Extreme Point/ Corner Point). *Let $S \subseteq \mathbf{R}^n$ be a convex set. A point $x^\star$ of S is called an extreme point or a corner point of S if $\nexists\, x, u$ $(x \neq u)$ in S, and $0 < \lambda < 1$ such that $x^\star = \lambda x + (1-\lambda)u$.*

Thus a point $x^\star$ is an extreme point of S if it does not lie on the line segment of any two distinct points of S. We may check that for the set $S_1 = \{(x_1, x_2) : x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 1\}$, the extreme points are (0,0), (1,0), and (0,1); whereas for the set $S_1 = \{(x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$, every point on the circle $x_1^2 + x_2^2 = 1$ is an extreme point.

Definition 3.2.4 (Hyperplane). *Let $p \subseteq \mathbf{R}^n$ and $d \in \mathbf{R}$. Then the set H defined as $H = \{x \in \mathbf{R}^n : p^T x = d\}$ is called a hyperplane.*

Thus a hyperplane H in $\mathbf{R}^n$ is the natural extension of line in $\mathbf{R}^2$ or a plane in $\mathbf{R}^3$. Also every hyperplane is a convex set.

Definition 3.2.5 (Closed Half Spaces). *Let $H = \{x \in \mathbf{R}^n : p^T x = d\}$ be a hyperplane in $\mathbf{R}^n$. Then the sets*

$$H_1 = \{x \in \mathbf{R}^n : p^T x \leq d\}$$

and

$$H_2 = \{x \in \mathbf{R}^n : p^T x \geq d\}$$

are called the closed half spaces generated by the hyperplane H.

We can check that H_1 and H_2 are convex sets in $\mathbf{R}^n$.

Definition 3.2.6 (Supporting Hyperplane). *Let $S \subseteq \mathbf{R}^n$ be a closed convex set. Let u be a boundary point of S. Then a hyperplane H is called a supporting hyperplane at u if it passes through u and whole of S is contained in one closed half spaces generated by H.*

Definition 3.2.7 (Edge). *Let $S \subseteq \mathbf{R}^n$ be a convex set, and $x, u \in S$ with $x \neq u$. Then the line segment joining x, u is called an edge of the convex set S if it is the intersection of S with a supporting hyperplane.*

Definition 3.2.8 (Adjacent Extreme Points). *Let $S \subseteq \mathbf{R}^n$ be a convex set. Then two extreme points $\overline{x}$ and $\overline{u}$ of S are called adjacent extreme points if they are joined by an edge.*

Definition 3.2.9 (Convex Combination). *Let $x, u \in \mathbf{R}^n$. Then the combination $\lambda x + (1-\lambda)u$, $0 \leq \lambda \leq 1$ is called the convex combination of x and u. In general, let $x^{(1)}, x^{(2)}, \ldots, x^{(r)}$ be r points in $\mathbf{R}^n$. Then the combination $\sum_{k=1}^{r} \lambda_k x^{(k)}$ with $\lambda_k \geq 0$ $(k = 1, 2, \ldots, r)$ and $\sum_{k=1}^{r} \lambda_k = 1$ is called the convex combination of r points $x^{(1)}, x^{(2)}, \ldots, x^{(r)}$.*

Here we must note the difference between the linear combination and convex combination. In linear combination $\alpha_1 x^{(1)} + \alpha_2 x^{(2)} + \ldots + \alpha_r x^{(r)}, \alpha_1, \alpha_2, \ldots, \alpha_r \in \mathbf{R}$. But in the convex combination the coefficient are non-negative and their sum equals one.

Definition 3.2.10 (Convex set spanned by a set). *Let $S \subseteq \mathbf{R}^n$. Then the set $CSpan(S)$ given by*

$$CSpan(S) = \left\{ \sum_{r=1}^{k} \lambda_r x^{(r)},\ \sum_{r=1}^{k} \lambda_r = 1,\ k \text{ finite (arbitrary)}, \lambda_r \geq 0, \text{ and } x^{(r)} \in S,\ \forall r \right\}$$

is called the convex set spanned by S.

Thus CSpan(S) is the set of all convex combinations of an arbitrary but finitely many elements of S. It is simple to check that CSpan(S) and Conv(S) are same. Also CSpan(S) is different from Span(S) as Span(S) is the set of all linear combinations of finitely many elements of S, and therefore it is a subspace of $\mathbf{R}^n$ where as CSpan(S) is a convex set in $\mathbf{R}^n$.

Definition 3.2.11 (Polyhedron/Polytope). *Let $S \subset \mathbf{R}^n$. Then S is called a polyhedron if it is the intersection of finite number of closed half spaces, i.e.*

$$S = \{x \in \mathbf{R}^n : p_i^T x \leq d_i \ (i = 1, \ldots, r)\}.$$

If a polyhedron is also bounded then it is called a *polytope*. A polytope is thus a closed, bounded, convex set having finitely many extreme points; the bounding surface being the hyperplanes. The set $\mathbf{R}^2_+$ given by $\mathbf{R}^2_+ = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 \geq 0, x_2 \geq 0\}$ is a polyhedron in $\mathbf{R}^2$ but not a polytope, where as the set $S = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0\}$ is a polytope in $\mathbf{R}^2$.

Definition 3.2.12 (Simplex in $\mathbf{R}^n$). *Let $x^{(1)}, x^{(2)}, \ldots, x^{(n)}, x^{(n+1)}$ be $(n+1)$ points in $\mathbf{R}^n$. Then the convex set spanned by these $(n+1)$ points is called a $n-$simplex in $\mathbf{R}^n$.*

A 2-simplex in $\mathbf{R}^2$ is a solid triangle and 3-simplex in $\mathbf{R}^3$ is a solid tetrahedron (the points inside the triangle/ tetrahedron are also being included).

It can also be checked that every point of the polytope can be expressed as a convex combination of its extreme points. Therefore if S is a polytope with extreme points $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$ then for any $x \in S$, there exist scalars $\alpha_1, \alpha_2, \ldots, \alpha_k$ such that $\alpha_r \geq 0$ $(r = 1, \ldots, k)$, $\sum_{r=1}^{k} \alpha_r = 1$ and $x = \sum_{r=1}^{k} \alpha_r x^{(r)}$.

3.3 Some Elementary Results for LPP

We consider the LPP

$$
\begin{aligned}
&\textit{Max} && z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
&\text{subject to} \\
& && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \left(\leq, =, \geq\right) b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \left(\leq, =, \geq\right) b_2 \\
& && \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \left(\leq, =, \geq\right) b_m \\
& && x_1 \geq 0, \ldots, x_n \geq 0\,, \qquad (3.1)
\end{aligned}
$$

where as explained in the last chapter, only one inequality sign holds in each constraint, though different constraints may have different inequality sign. We denote by S, the feasible region of LPP (3.1).

Result 3.3.1 *The feasible region S is a convex subset of $\mathbf{R}^n$.*

Proof. Let S_i denote the set of all points $x \in \mathbf{R}^n$ for which the ith constraint of LPP (3.1) holds $(i = 1, 2, \ldots, m, m+1, \ldots, m+n)$. Then each S_i is either a hyperplane or one of the closed halfspaces, and hence a convex set. But $S = \cap_i S_i$ is the intersection of finitely many convex sets, consequently it is a convex set. □

Result 3.3.2 *Let LPP (3.1) has an optimal solution $x^\star$. Then $x^\star$ can not be in the interior of the feasible region S.*

Proof. Since $x^\star$ is optimal to problem (3.1), we have $c^T x^\star \geq c^T x,\ \forall\ x \in S$. If possible let $x^\star \in intS$. Then by the definition of interior point there exists a neighborhood $N_\epsilon(x^\star) \subset S$. Now we define

$$\overline{x} = x^\star + \frac{\epsilon}{2}\frac{c}{\|c\|}\,. \qquad (3.2)$$

Then $\|\overline{x} - x^\star\| = \frac{\epsilon}{2} < \epsilon$ and hence $\overline{x} \in N_\epsilon(x^\star) \subset S$, i.e. $\overline{x}$ is certainly feasible for the given problem. But

$$
\begin{aligned}
c^T\bar{x} &= c^T\left(x^\star + \frac{\epsilon}{2}\frac{c}{\|c\|}\right) \\
&= c^T x^\star + \frac{\epsilon}{2}\frac{c^T c}{\|c\|} \\
&= c^T x^\star + \frac{\epsilon}{2}\frac{\|c\|^2}{\|c\|} \\
&= c^T x^\star + \frac{\epsilon}{2}\|c\| \,. \qquad (3.3)
\end{aligned}
$$

Therefore (3.3) gives $(c^T\bar{x} - c^T x^\star) = \frac{\epsilon}{2}\|c\| > 0$, which contradicts that $x^\star$ is optimal to (3.1). Hence $x^\star \notin intS$. □

Result 3.3.3 *If the given LPP has an optimal solution then at least one corner point of S is optimal.*

Proof. Although the proof holds in much more generality, we shall give the proof for the case when S is a polytope. In this case, it is guaranteed that the given LPP has an optimal solution, so that we have to only show that the optimal value is certainly attained at an extreme point.

Let $x^\star$ be an optimal point of the given LPP. If $x^\star$ is an extreme point then the result holds obviously. So we consider the case when $x^\star$ is not an extreme point. Then, since S is a polytope it has finitely many corner points, say $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$, and any point of S can be expressed as a convex combination of the corner points of S. In particular this holds for $x^\star$ as well, i.e. there exist scalars $\alpha_1^\star, \alpha_2^\star, \ldots, \alpha_k^\star$ such that

$$
x^\star = \sum_{r=1}^{k} \alpha_r^\star x^{(r)}, \quad \sum_{r=1}^{k} \alpha_r^\star = 1, \quad \alpha_r^\star \geq 0, \forall\, r. \qquad (3.4)
$$

Therefore,

$$
c^T x^\star = c^T\left(\textstyle\sum_{r=1}^{k} \alpha_r^\star x^{(r)}\right) = \sum_{r=1}^{k} \alpha_r^\star \left(c^T x^{(r)}\right),
$$

i.e. $c^T x^\star$ is the weighted arithmetic mean (with weights $\alpha_r^\star$) of k scalars $c^T x^{(r)}$. Hence by the property of the arithmetic mean,

$$
\begin{aligned}
c^T x^\star &= \sum_{r=1}^{k} \alpha_r^\star (c^T x^{(r)}) \\
&\leq Max(c^T x^{(1)}, \ldots, c^T x^{(k)}) \\
&= c^T x^{(p)} \text{ for some } 1 \leq p \leq k. \qquad (3.5)
\end{aligned}
$$

But $x^\star$ is optimal, so $c^T x^\star \geq c^T x$, $\forall\, x \in S$. In particular

$$
c^T x^\star \geq c^T x^{(p)} \,. \qquad (3.6)
$$

Equations (3.5) and (3.6) give $c^T x^\star = c^T x^{(p)}$, which proves the result. This is because if $x^\star$ is not an extreme point then there exists an extreme point $x^{(p)}$ which is also optimal. Thus at least one extreme point is certainly optimal. □

Result 3.3.4 *Every local optimal point of LPP (3.1) is also a global optimal point.*

Proof. Let $\bar{x}$ be a local max point of LPP (3.1). Then by the definition of local max point, there exists a neighborhood $N_\delta(\bar{x})$ such that $c^T\bar{x} \geq c^T x$ for all $x \in N_\delta(\bar{x}) \cap S$, where S is the feasible region of the given LPP. (We have taken $N_\delta(\bar{x}) \cap S$ here because the $\delta-$ neighborhood of $\bar{x}$ is to be considered in the relative topology of S). Let u be any arbitrary point outside $N_\delta(\bar{x}) \cap S$. The result will be proved if we can show that $c^T\bar{x} \geq c^T u$.

Now we refer to the below given figure (Fig 3.3) and note that there certainly exists a point $\hat{x} \in N_\delta(\bar{x}) \cap S$ and $0 < \lambda < 1$ such that $\hat{x} = \lambda u + (1-\lambda)\bar{x}$.

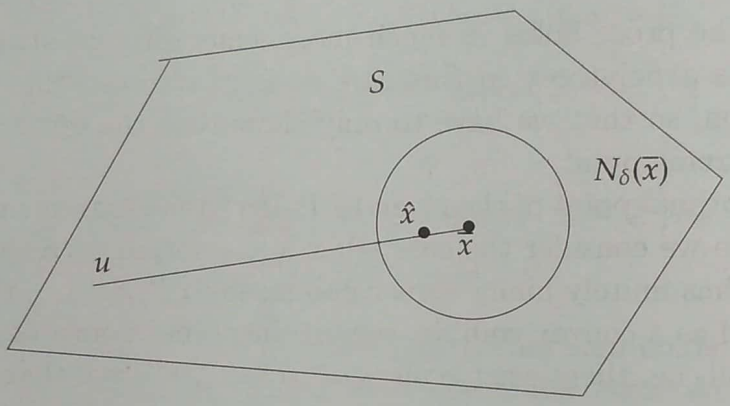


Fig. 3.3.

Therefore

$$\begin{aligned} c^T\hat{x} &= c^T(\lambda u + (1-\lambda)\bar{x}) \\ &= \lambda(c^T u) + (1-\lambda)(c^T\bar{x}). \end{aligned} \tag{3.7}$$

But as $\bar{x}$ is a local max point and $\hat{x} \in N_\delta(\bar{x}) \cap S$, we have $c^T\bar{x} \geq c^T\hat{x}$. Therefore equation (3.7) gives

$$\begin{aligned} \lambda(c^T u) + (1-\lambda)(c^T\bar{x}) &= c^T\hat{x} \leq c^T\bar{x}, \\ \text{i.e. } \lambda(c^T u) &= (c^T\bar{x}) - (1-\lambda)(c^T\bar{x}), \\ \text{i.e. } \lambda(c^T u) &= \lambda(c^T\bar{x}), \end{aligned}$$

which because of $0 < \lambda < 1$, gives $c^T u = c^T\bar{x}$. Hence $\bar{x}$ is a global max point. □

Result 3.3.5 *The set of all optimal solutions of a LPP is a convex set.*

Proof. If a LPP has no optimal solution or has only one optimal solution then the result is true by convention. So let us assume that the given LPP has at least two optimal solutions. Let V denote the set of all optimal solutions of the given LPP, i.e.

$$V = \{x \in S : x \text{ is optimal}\},$$

where S is the feasible region of the given LPP. We shall prove that V is a convex set.

Let $x^{(1)}$ and $x^{(2)} \in V$. Let $\hat{x} = \lambda x^{(1)} + (1-\lambda)x^{(2)}, 0 \leq \lambda \leq 1$. Then $\hat{x} \in S$ because $x^{(1)} \in S$, $x^{(2)} \in S$ and S is a convex set. Also $c^T x^{(1)} \geq c^T x$ for all $x \in S$ and $c^T x^{(2)} \geq c^T x$ for all $x \in S$. Hence $\hat{x} \in V$, which gives that V is a convex set. As a consequence of Result 3.3.5 we get that if a LPP has more than one optimal solution, then it has infinitely many optimal solutions. □

Remark 3.3.1 *Though the above results have been proved for the LPP (3.1) which is in the maximization form, these results hold for LPP's in the minimization form as well.*

In view of the above discussions, we observe that because of the structure of linearity on LPP's, the below given properties are guaranteed. Further, only because of these properties, we succeeded in developing a method like the simplex method to solve LPP's. These properties are

(P1) The feasible region of LPP is always a convex set. Infact it is polyhedron/polytope.
(P2) If the given LPP has an optimal solution then at least one corner point (extreme point) of the feasible region is optimal.
(P3) For LPP's, every local max point is a global max point. Also every local min point is a global min point.

Thus properties (P1), (P2) and (P3), are very basic to the algorithmic study of LPP's and the main reason for having them is the presence of the structure of linearity. For nonlinear programming problems the structure of linearity is missing and therefore, in general, we can not guarantee these properties and that makes NLP's much more difficult to solve. Following examples illustrate these points.

Example 3.3.1 *Consider the optimization problem*

$$\begin{aligned} \textit{Max} \quad & z = 4x_1 + 3x_2 \\ \textit{subject to} \quad & \\ & x_1 + x_2 \leq 3 \\ & x_1 x_2 \leq 1 \\ & x_1, x_2 \geq 0. \end{aligned}$$

and check if its feasible region is a convex set.

Solution The given optimization problem is a NLP as the constraint $x_1 x_2 - 1 \leq 0$ is a nonlinear function of decision variables x_1 and x_2. Here the feasible region is depicted in Fig 3.4 which is clearly not a convex set.

Therefore there is no guarantee that the property (P1) holds for NLP's.

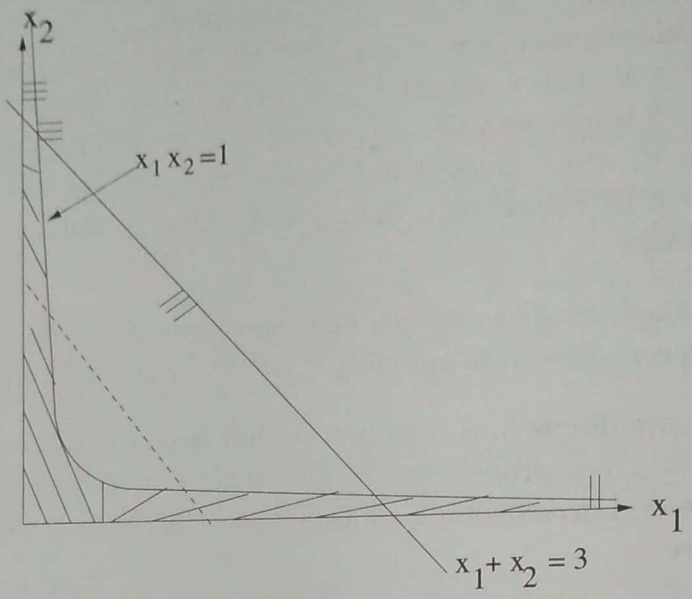


Fig. 3.4.

Example 3.3.2 *Consider the optimization problem*

$$\begin{aligned} &\textit{Min} && z = (x_1 - 3)^2 + (x_2 - 3)^2 \\ &\textit{subject to} \\ & && x_1 + x_2 \leq 4 \\ & && x_1 - x_2 \leq 2 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

Is the optimal point a corner point?

Solution The given optimization problem is a (NLP) as the objective function is a nonlinear function of decision variables x_1 and x_2. Here the feasible region is a polytope with corner points O, A, B, and C; and the optimal solution of the given problem is the point P, which is not a corner point as depicted in Fig. 3.5.

Therefore there is no guarantee that the property (P2) holds for NLP's. Infact if in the above example we change the objective function to $z = (x_1 - 2)^2 + (x_2 - 1)^2$, then the optimal point (2,1) lies in the interior of the feasible region S.

Example 3.3.3 *Consider the optimization problem of minimizing the function* $f(x) = (|x| - 10)cos(2\pi x)$ *over* $[-10, 10]$. *Is a local min point also global min point?*

Solution The function $f(x) = (|x| - 10)cos(2\pi x)$ is called one dimensional wave function. Its graph over [-10,10] is given in Fig 3.6. This function has many local min points but $x^\star = 0$ is the global min point.

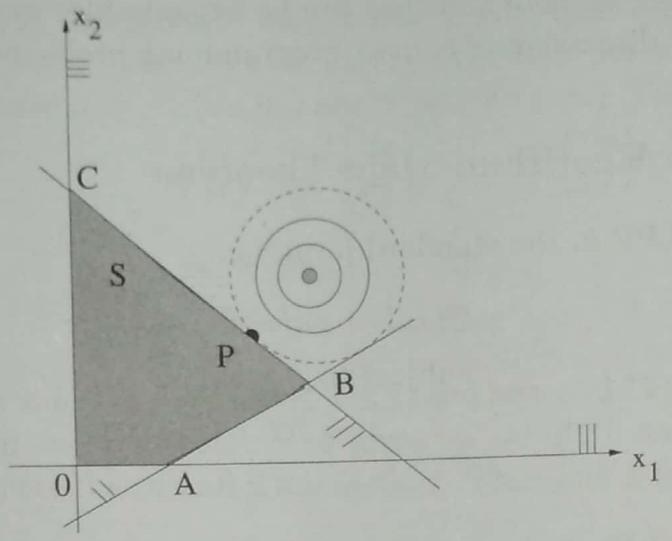


Fig. 3.5.

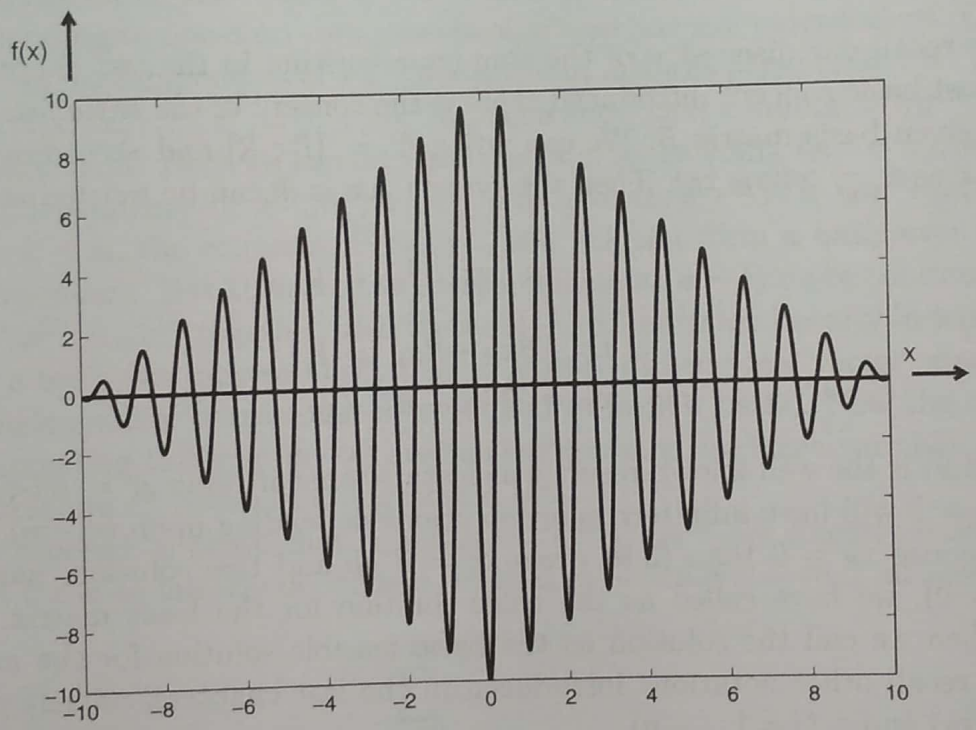


Fig. 3.6.

There are many other difficult global optimization benchmark problems available in the literature. It is always a challenge to obtain the global min point because most of the standard nonlinear programming algorithms give only the local min point. We shall discuss some aspects of global optimization algorithms in the later chapters. Though we can not guarantee properties (P1)-(P3) for a general NLP, there is a class of NLP's, called convex programming problems for which some of these properties can be guaranteed to hold. This is because of the fact that although the structure of linearity is missing,

the convexity structure is there and that can be exploited for studying such NLP's. We shall have a detailed discussion of convex programming problems later in the book.

3.4 The Simplex Algorithm: Main Theorems

Let us consider the LPP in the standard form, i.e.

$$\begin{aligned} &\text{Max} && z = c^T x \\ &\text{subject to} \\ & && Ax = b \\ & && x \geq 0, \end{aligned} \tag{3.8}$$

with (i) $b \geq 0$ and (ii) Rank $A = m(< n)$. Here $x \in \mathbf{R}^n$, $c \in \mathbf{R}^n$, $b \in \mathbf{R}^n$ and $A = [a_{ij}]$ is an $(m \times n)$ matrix. Also $S = \{x \in \mathbf{R}^n : Ax = b, x \geq 0\}$ is the feasible region of LPP (3.8).

We now recall our discussion of the simplex algorithm in the last chapter and note that the most basic concept introduced there is the concept of the *basic feasible solution* x_B for the given basis matrix B. We can write $A = [B : R]$ and accordingly partition the vector x as $x = col(x_B, x_R)$. Then the system $Ax = b$ can be written as

$$\begin{aligned} [B : R]\begin{bmatrix} x_B \\ x_R \end{bmatrix} &= b, \\ i.e. \quad Bx_B + Rx_R &= b, \\ i.e. \quad x_B &= B^{-1}b - B^{-1}Rx_R. \end{aligned} \tag{3.9}$$

Equation (3.9) is the well known result which states that if Rank $A = m(< n)$ then the system $Ax = b$ will have infinitely many solutions depending upon $(n - m)$ parameters x_R. If we choose $x_R = 0$ then (3.9) gives $x_B = B^{-1}b$ and this solution, namely, $(x_B = B^{-1}b,\ x_R = 0)$, we have called as the basic solution for the basis matrix B. Further if $x_B \geq 0$ then we call the solution as the basic feasible solution for the basis matrix B. We also recall other notations introduced in the last chapters, namely c_B, $y^{(j)}$ $(j = 1, \ldots, n)$, $z(x_B)$ and z_j $(j = 1, \ldots, n)$.

We now have the following main theorems, which have been used in the last chapter for the development of the simplex algorithm.

Theorem 3.4.1 *Every extreme point of the set S is a b.f.s to the system of equations $Ax = b, x \geq 0$ and conversely every b.f.s of the above system is an extreme point of the set S.*

Proof. (i) Let x be a b.f.s of the system $Ax = b, x \geq 0$. We wish to prove that x is an extreme point of the set S. As x is a b.f.s., it should be a b.f.s for a given basis matrix B and therefore we can write $x = (x_B = B^{-1}b,\ x_R = 0)$. If possible, let x be not an extreme

point of S. Then, by the definition of an extreme point, this implies that there exist $u \in S, v \in S, u \neq v$, $0 < \lambda < 1$ such that $x = \lambda u + (1-\lambda)v$. Now partitioning u and v as per the partition of x, we have $u = col(u_B, u_R)$ and $v = col(v_B, v_R)$. Then $x = \lambda u + (1-\lambda)v$ gives

$$\begin{pmatrix} x_B \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} u_B \\ u_R \end{pmatrix} + (1-\lambda) \begin{pmatrix} v_B \\ v_R \end{pmatrix}$$

$$i.e.\ x_B = \lambda u_B + (1-\lambda) v_B \tag{3.10}$$

$$0 = \lambda u_R + (1-\lambda) v_R \tag{3.11}$$

But $Au = b$, $u \geq 0$; $Av = b$, $v \geq 0$. Also $0 \leq \lambda \leq 1$, and hence (3.11) gives $u_R = 0 = v_R$. This together with $Au = b$ and $Av = b$ gives $u_B = B^{-1}b$ and $v_B = B^{-1}b$. Thus $u = v$, which contradicts that u and v are distinct. Therefore x is an extreme point of S.

(ii) Next let $x^\star = col(x_1, x_2, \ldots, x_n)$ be an extreme point of S. We shall prove that $x^\star$ is a b.f.s to the system $Ax = b$, $x \geq 0$. For this, it is enough to show that columns $a^{(j)}$ of A corresponding to non-zero components of $x^\star$ are linearly independent. Let k be the number of nonzero components of $x^\star$. Then without any loss of generality we can assume that $x^\star = col(x_1^\star, x_2^\star, \ldots, x_k^\star, 0, 0 \ldots, 0)$, and then show that columns $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ are linearly independent. Here we should note that $k \leq m$, as Rank $(A) = m$. For $k = m$, the linear independence of $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ will give that $x^\star$ is a nondegenerate b.f.s. However for $k < m$, the columns $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ will not form a basis even if they are linearly independent. But then we can always augment $(m-k)$ more columns such that columns $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ together with these $(m-k)$ columns are linearly independent and hence form a basis (we may recollect that in a finite dimensional vector space every set of linearly independent vectors can be extended to form a basis). Now the components of $x^\star$ corresponding to these $(m-k)$ augmented columns are basic variables at the zero level and therefore $x^\star$ becomes a degenerate b.f.s.

Now we proceed to prove that columns $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ are linearly independent. If possible, let these be linearly dependent, so that exist scalars λ_i (not all zero) such that

$$\sum_{i=1}^{k} \lambda_i a_i = 0\,. \tag{3.12}$$

Also $Ax^\star = b$, $x^\star \geq 0$. Therefore

$$\sum_{i=1}^{k} x_i^\star a^{(i)} = b,\ x_i^\star \geq 0\ (i = 1, \ldots, k). \tag{3.13}$$

We now define $\eta = \underset{i}{Min} \left\{ \dfrac{x_i^\star}{|\lambda_i|},\ \lambda_i \neq 0 \right\}$, and choose $\epsilon > 0$ such that $0 < \epsilon < \eta$. This gives $(x_i^\star + \epsilon\lambda_i) > 0$ $(i = 1, \ldots, k)$ and $(x_i - \epsilon\lambda_i) > 0$ $(i = 1, \ldots, k)$. Now take

$\lambda = col(\lambda_1, \lambda_2, \ldots, \lambda_k, 0, 0, \ldots, 0) \in \mathbf{R}^n$, $x^{(1)} = x^\star + \epsilon\lambda$, and $x^{(2)} = x^\star - \epsilon\lambda$. Then $x^{(1)} \geq 0$ and $x^{(2)} \geq 0$. Also because of (3.12) and (3.13), $Ax^{(1)} = b$, $Ax^{(2)} = b$. Therefore $x^{(1)} \in S$ and $x^{(2)} \in S$. But $x^\star = (x^{(1)} + x^{(2)})/2$, which contradicts that $x^\star$ is an extreme point of S. Hence columns $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ are linearly independent as desired. This prove that $x^\star$ is a b.f.s of the system $Ax = b, x \geq 0$. □

Let us now recall the simplex tableau

	$y^{(1)}$ $y^{(2)}$	$y^{(j)}$	... ... $y^{(n)}$
x_{B1}		y_{1j}	
x_{B2}		y_{2j}	
$\vdots$		$\vdots$	
x_{Bm}		y_{mj}	
$z(x_B)$		$(z_j - c_j)$	

and associated results which have been used in the stepwise description of the simplex method. We shall prove these results now.

Theorem 3.4.2 *If some $z_j - c_j < 0$ and for that j some $y_{ij} > 0$, then there exists a new b.f.s. $\hat{x}_B$ such that $z(\hat{x}_B) \geq z(x_B)$.*

Proof. Let the hypotheses of the theorem hold for the current b.f.s. x_B corresponding to the basis matrix $B = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, b^{(r)}, b^{(r+1)}, \ldots, b^{(m)}]$. Let B be changed to $\hat{B}$, where

$$\hat{B} = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, a^{(j)}, b^{(r+1)}, \ldots, b^{(m)}]$$

i.e. from B, column $b^{(r)}$ has been taken out and at that place column $a^{(j)}$ has been entered. So far we have not put any conditions on j and r but obviously there must be some conditions on j and r such that $\hat{B}$ gives a b.f.s. $\hat{x}_B$ such that $z(\hat{x}_B) \geq z(x_B)$. Then to prove the theorem, we have to show that the conditions on j and r as chosen here, shall be met under the hypotheses of the theorem.

Now the first thing we require is that columns of $\hat{B}$ are linearly independent because only then $\hat{x}_B$ will be a basic solution. Next we wish that every component $\hat{x}_{B_i}$ of $\hat{x}_B$ is non-negative so that it is a b.f.s.; and finally this $\hat{x}_B$ should be such that $z(\hat{x}_B) \geq z(x_B)$. We now consider all of these one by one.

Since $a^{(j)}$ is a non-basic column and columns of B form a basis for $\mathbf{R}^m$, we have

$$a^{(j)} = y_{1j}b^{(1)} + \ldots + y_{rj}b^{(r)} + \ldots + y_{mj}b^{(m)}. \tag{3.14}$$

Then if $b^{(r)}$ is replaced by $a^{(j)}$ to get the new matrix $\hat{B}$ and we wish that columns of $\hat{B}$ are linearly independent, then by the *replacement theorem of vector spaces*, we must have $y_{rj} \neq 0$. (Let V be a vector space of dimension n with the basis as $v_1, v_2, \ldots, v_n$. For $u \in V$, we have $u = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_r v_r + \ldots + \alpha_n v_n$. The Replacement Theorem states that if $\alpha_r \neq 0$ then $(v_1, \ldots, v_{r-1}, u, v_{r+1}, \ldots, v_n)$ is also a basis of V). So the first condition

on r and j is that $y_{rj} \neq 0$ and this gives that columns of $\widehat{B}$ are linearly independent and hence it is a basis matrix which gives the basic solution $\hat{x}_B$. Now we attempt to find $\hat{x}_B$ explicitly. For this we note that $x_B = B^{-1}b$, i.e. $Bx_B = b$ or in terms of components.

$$\sum_{i=1}^{m} x_{B_i} b^{(i)} = b, \tag{3.15}$$

i.e.

$$\sum_{\substack{i=1,\\ i\neq r}}^{m} x_{B_i} b^{(i)} + x_{B_r} b^{(r)} = b.$$

But as $y_{rj} \neq 0$, (3.14) gives

$$b^{(r)} = \frac{1}{y_{rj}} \left[a^{(j)} - \sum_{\substack{i=1,\\ i\neq r}}^{m} y_{ij} b^{(i)} \right] \tag{3.16}$$

which on substitution in (3.15) gives

$$\sum_{\substack{i=1,\\ i\neq r}}^{m} x_{B_i} b^{(i)} + \frac{x_{B_r}}{y_{r_j}} \left[a^{(j)} - \sum_{\substack{i=1,\\ i\neq r}}^{m} y_{ij} b^{(i)} \right] = b. \tag{3.17}$$

i.e.

$$\sum_{\substack{i=1,\\ i\neq r}}^{m} \left(x_{B_i} - \frac{x_{B_r} y_{ij}}{y_{r_j}} \right) b^{(i)} + \frac{x_{B_r}}{y_{r_j}} a^{(j)} = b. \tag{3.18}$$

Therefore if we define $\hat{x}_B$ as

$$\hat{x}_{B_i} = \begin{cases} x_{B_i} - \dfrac{x_{B_r} y_{ij}}{y_{rj}}, & (i \neq r) \\ \dfrac{x_{B_r}}{y_{rj}}, & (i = r) \end{cases} \tag{3.19}$$

then (3.18) becomes $\hat{B}\hat{x}_B = b$, i.e. $\hat{x}_B = (\hat{B})^{-1}b$, is the basic solution whose components are given in (3.19).

Having obtained the basic solution $\hat{x}_B$ (for that we needed condition on r and j so that $y_{rj} \neq 0$) for the new basic matrix $\hat{B}$, we have now to put additional conditions on r and j so that $\hat{x}_B$ becomes basic feasible. For this we need that all components of $\hat{x}_B$ should be non-negative. Now looking at equation (3.19), we note that for $\hat{x}_{B_r}$ to be non-negative we need $y_{rj} > 0$. For $i \neq r$, component $\hat{x}_{B_i}$ is certainly non-negative for those i for which $y_{ij} \leq 0$. So we have to bother only when $y_{ij} > 0$. Thus we want that for $y_{ij} > 0$ as well

$$x_{B_i} - \frac{x_{B_r}}{y_{rj}} y_{ij} \geq 0,$$

i.e.

$$\frac{x_{B_i}}{y_{ij}} - \frac{x_{B_r}}{y_{rj}} \geq 0,$$

i.e.

$$\frac{x_{B_r}}{y_{rj}} = \underset{i}{Min} \left\{ \frac{x_{B_i}}{y_{ij}} : y_{ij} > 0 \right\}. \tag{3.20}$$

The relation (3.20) is called the *minimum ratio criteria* and it guarantees that irrespective of j, as long as $y_{rj} > 0$, if we chose r as per (3.20) then the next solution $\hat{x}_B$ is certainly basic feasible.

Next we wish to put condition on j (note that condition on r has already been fixed now as per equation (3.20)) so that $z(\hat{x}_B) \geq z(x_B)$. For this we note that $z(x_B) = c_B^T x_B = \sum_{i=1}^{m} c_{B_i} x_{B_i}$ and

$$\begin{aligned}
z(\hat{x}_B) &= \sum_{i=1}^{m} \hat{c}_{B_i} \hat{x}_{B_i} \\
&= \sum_{\substack{i=1, \\ i \neq r}}^{m} c_{B_i} \left(x_{B_i} - \frac{x_{B_r}}{y_{rj}} y_{ij} \right) + \hat{c}_{B_r} \frac{x_{B_r}}{y_{rj}} \\
&= \sum_{i=1}^{m} c_{B_i} \left(x_{B_i} - \frac{x_{B_r}}{y_{rj}} y_{ij} \right) + c_j \frac{x_{B_r}}{y_{rj}} \qquad (3.21) \\
&= \sum_{i=1}^{m} c_{B_i} x_{B_i} - \frac{x_{B_r}}{y_{rj}} \left(\sum_{i=1}^{m} c_{B_i} y_{ij} - c_j \right) \\
&= z(x_B) - \frac{x_{B_r}}{y_{rj}} (z_j - c_j)
\end{aligned}$$

i.e.

$$z(\hat{x}_B) = z(x_B) - \frac{x_{B_r}}{y_{rj}} (z_j - c_j). \tag{3.22}$$

Here, in (3.21), $\hat{c}_{B_r} = c_j$ as the entering column is $a^{(j)}$ and for that the corresponding basic variable is x_j whose coefficient in the objective function is c_j. For $i \neq r$, the basic columns have not changed and so $c^*_{B_r} = c_{B_i}$. Also, including $i = r$ in the summation amounts to adding a 'zero' value in (3.21) and hence there is no change.

Now if we chose our j such that $(z_j - c_j) < 0$ then from (3.22), $z(\hat{x}_B) \geq z(x_B)$. Thus we must have *some* j for which $(z_j - c_j) < 0$ and for that *some* $y_{ij} > 0$. Then we can chose r as from (3.20) and get a new b.f.s. $\hat{x}_B$ with $z(\hat{x}_B) \geq z(x_B)$. This proves the theorem. □

Theorem 3.4.3 *If all* $(z_j - c_j) \geq 0$ *then the current b.f.s.* x_B *is optimal to the given LPP.*

Proof. Let $x \in S$ be arbitrary point. We have to show that under the condition $(z_j - c_j) \geq 0$, $\forall\ j$, $z(x_B) \geq z(x)$, where $z(x_B) = c_B^T x_B$ and $z(x) = c^T x$. For this we start with $z_j - c_j \geq 0$, $\forall\ j$, and as $x_j \geq 0$, $\forall\ j$, we have

$$z_j x_j \geq c_j x_j,\ \forall\ j.$$

Now summing over j, we get

$$\sum_{j=1}^{n} z_j x_j \geq \sum_{j=1}^{n} c_j x_j$$

i.e.

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} c_{B_i} y_{ij} \right) x_j \geq c^T x.$$

But on LHS these are finite sums and so we can interchange the order and get

$$\sum_{i=1}^{m} c_{B_i} \left(\sum_{j=1}^{n} y_{ij} x_j \right) \geq c^T x\,. \tag{3.23}$$

From (3.23) we observe that the theorem will be proved if we can show that

$$x_{B_i} = \sum_{j=1}^{n} y_{ij} x_j, \tag{3.24}$$

because then the LHS of (3.23) will be $c_B^T x_B$ which equals $z(x_B)$.
Now to prove (3.23) we note that $Ax = b$ and hence

$$\begin{aligned}
&\text{i.e.} \quad x_B = B^{-1}b = B^{-1}Ax,\\
&\text{i.e.} \quad x_B = B^{-1}[a^{(1)}, a^{(2)}, \ldots, a^{(j)}, \ldots, a^{(n)}]x,\\
&\text{i.e.} \quad x_B = [B^{-1}a^{(1)}, B^{-1}a^{(2)}, \ldots, B^{-1}a^{(j)}, \ldots, B^{-1}a^{(n)}]x,\\
&\text{i.e.} \quad x_B = [y^{(1)}, y^{(2)}, \ldots, y^{(j)}, \ldots, y^{(n)}]x\,.
\end{aligned} \tag{3.25}$$

But (3.25) is a vector equality and therefore we have to equate componentwise. The ith component on LHS is $y_{i_1}x_1 + y_{i_2}x_2 + \ldots + y_{i_n}x_n$. Therefore $x_{B_i} = \sum_{j=1}^{n} y_{ij}x_j$ as desired. □

Theorem 3.4.4 *If some $(z_j - c_j) < 0$ and for that j all $y_{ij} \leq 0$, then the given LPP has unbounded solution.*

Proof. Let us choose that j for which the hypotheses of the theorem hold, i.e. $(z_j - c_j) < 0$ and $y_{ij} \leq 0$ for all $i = 1, \ldots, m$. Now for the current b.f.s. x_B,

$$\sum_{i=1}^{m} x_{B_i} b^{(i)} = b \,. \tag{3.26}$$

If we choose $\theta \in \mathbf{R}$ arbitrary then (3.26) can be rewritten as

$$\sum_{i=1}^{m} x_{B_i} b^{(i)} + \theta a^{(j)} - \theta a^{(j)} = b \,. \tag{3.27}$$

But $a^{(j)}$ is a non-basic column and therefore

$$a^{(j)} = \sum_{i=1}^{m} y_{ij} b^{(i)} \,. \tag{3.28}$$

Substitution for $a^{(j)}$ from (3.28) into (3.27) we get

$$\sum_{i=1}^{m} x_{B_i} b^{(i)} + \theta a^{(j)} - \theta\Big(\sum_{i=1}^{m} y_{ij} b^{(i)}\Big) = b,$$

i.e.

$$\sum_{i=1}^{m} (x_{B_i} - \theta y_{ij}) b^{(i)} + \theta a^{(j)} = b. \tag{3.29}$$

If we now define a vector $\hat{x}$ as $\hat{x} = col(\hat{x}_1, \ldots, \hat{x}_m, \hat{x}_{m+1}, 0, 0, \ldots, 0)$ where

$$\hat{x}_i = x_{B_i} - \theta y_{ij} \; (i = 1, \ldots, m) \text{ and } \hat{x}_{m+1} = \theta. \tag{3.30}$$

Then (3.29) gives

$$A\hat{x} = b.$$

If we further choose $\theta > 0$, then $\hat{x} \geq 0$ as $y_{ij} \leq 0$, $\forall\, i$. Therefore using the condition $y_{ij} \leq 0$, $\forall\, i$, and knowing the current b.f.s. x_B, we have been able to construct a feasible solution $\hat{x}$ as described above. Here we may note that $\hat{x}$ is feasible but NOT basic feasible because in (3.29) there are $m + 1$ columns, namely $b_1, b_2, \ldots, b_m$ and also $a^{(j)}$.

So far we have not used the condition $(z_j - c_j) < 0$. For this, let us find the value of the objective function for $\hat{x}$, i.e.

$$\begin{aligned} z(\hat{x}) &= \sum_{i=1}^{m} c_{B_i}\hat{x}_i + \hat{x}_{m+1}(c_j) \\ &= \sum_{i=1}^{m} c_{B_i}(x_{B_i} - \theta y_{ij}) + \theta c_j \\ &= \sum_{i=1}^{m} c_{B_i}x_{B_i} - \theta\Big(\sum_{i=1}^{m} c_{B_i}y_{ij} - c_j\Big), \end{aligned}$$

i.e.

$$z(\hat{x}) = z(x_B) - \theta(z_j - c_j)\,. \tag{3.31}$$

Since $(z_j - c_j) < 0$ and $\theta > 0$ is arbitrary, equation (3.31) tells that the objective function value $z(\hat{x})$ can be made arbitrary large by choosing $\theta > 0$ arbitrary large. This proves that the given LPP has unbounded solution. □

One important point to be noted in the above proof is that it not only shows that the given LPP has unbounded solution but also constructs the feasible point $\hat{x}$ for which the arbitrary large chosen value $z(\hat{x})$ of the objective function will be attained. The following example is illustrative in this context.

Example 3.4.1 *Use the simplex method to verify that the following LPP has unbounded solution*

$$\begin{aligned} \textit{Max} \quad & z = 2x_1 + 3x_2 \\ \textit{subject to} \quad & \\ & x_1 - x_2 \le 2 \\ & -3x_1 + x_2 \le 4 \\ & x_1,\ x_2 \ge 0. \end{aligned}$$

Hence obtain a feasible solution for which the objective function takes the value 496.

Solution Introducing the slack variables x_3 and x_4 and solving the problem as usual by the simplex method we get the following tableaus

		$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_3 = 2$	1	-1	1	0
←	$x_4 = 4$	-3	[1]	0	1
	0	-2	-3	0	0
		↑			

	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 6$	-2	0	1	1
$x_2 = 4$	-3	1	0	1
12	-11	0	0	3

Now Theorem 3.4.4 is applicable which confirms that the given LPP has unbounded solution. Next we have to find a feasible solution $\hat{x}$ for which the objective function attains the value 496. In terms of our notations $z(\hat{x}) = 496$, $z(x_B) = 12$ and $(z_1 - c_1) = -11$. Therefore using the relation (3.31) we get

$$496 = 12 - \theta(-11)$$

i.e.

$$\theta = 484/11 = 44.$$

Now we use (3.30) to construct $\hat{x}$. For this we note that here x_3 and x_2 are basic variables and $j = 1$. Hence

$$\hat{x} = \begin{pmatrix} x_1^* = 44 \\ x_2^* = 4 - 44(-3) = 136 \\ x_3^* = 6 - 44(-2) = 94 \\ x_4^* = 0 \end{pmatrix}$$

is the desired feasible solution.

Corollary 3.4.1. *If all $(z_j - c_j) \geq 0$ and there exists a j such that $y_{ij} \leq 0$ for all $i = 1, \ldots, m$, then the given LPP has unbounded feasible region but bounded optimal solution.*

Proof. As $(z_j - c_j) \geq 0$ for all j, by Theorem 3.4.3, the current solution x_B is optimal, so the given LPP has bounded optimal solution.

The feasible region is unbounded follows from the construction of $\hat{x}$ in the proof of Theorem 3.4.4. □

Theorem 3.4.5 *If in the optimal simplex tableau, for some non-basic variable x_j, $z_j - c_j = 0$ and for that some $y_{ij} > 0$, then the given LPP has infinitely many optimal solutions.*

Proof. We already have one optimal solution, namely the current one. Therefore the theorem will be proved if we could provide one more optimal solution and then use the fact that the set of all optimal solutions for a given LPP is a convex set. For this, choose that j for which the hypothesis of the theorem holds. As some $y_{ij} > 0$, we can certainly enter x_j and do one more iteration by choosing the variable to leave as per the minimum ratio criteria. This will give another basic feasible solution $\hat{x}_B$ which will be optimal because $z(\hat{x}_B) = z(x_B)$. This follows due to the fact that

$$z(\hat{x}_B) = z(x_B) - \frac{x_{B_r}}{y_{rj}}(z_j - c_j),$$

and $(z_j - c_j) = 0$. This proves the theorem. □

Theorem 3.4.6 *Let $B = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, b^{(r)}, \ldots, b^{(m)}]$ be the current b.f.s. Let column to enter $a^{(j)}$ and column to leave $b^{(r)}$ be determined as per the criteria of Theorem 3.4.2. Then for the basis matrix $\hat{B} = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, a^{(j)}, \ldots, b_m]$, the new b.f.s. $\hat{x}_B$, new objective function value $z(\hat{x}_B)$, new vector $\hat{y}^{(k)}$ and new values of $(\hat{z}_k - c_k)$ are given by*

$$(i)\quad \hat{x}_{B_i} = \begin{cases} x_{B_i} - \dfrac{x_{B_r} y_{ij}}{y_{rj}}, & (i \neq r) \\ \dfrac{x_{B_r}}{y_{rj}}, & (i = r) \end{cases}$$

$$(ii)\quad z(\hat{x}_B) = z(x_B) - \frac{x_{Br}}{y_{rj}}(z_j - c_j)$$

$$(iii)\quad \hat{y}_{ik} = \begin{cases} y_{ik} - \dfrac{y_{ij} y_{rk}}{y_{rj}}, & (i \neq r) \\ \dfrac{y_{rk}}{y_{rj}}, & (i = r) \end{cases}$$

$$(iv)\quad (\hat{z}_k - c_k) = (z_k - c_k) - \frac{y_{rk}}{y_{rj}}(z_j - c_j)$$

Proof. Here we must note that proving of above four relations is equivalent to saying that pivoting is justified. We have already proved (as part of the proof of Theorem 3.4.2) the first two relations and therefore we shall prove relations (iii) and (iv) only.

Let us go back to the proof of Theorem (3.4.2) and observe that

$$b^{(r)} = \frac{1}{y_{rj}}\left(a^{(j)} - \sum_{\substack{i=1,\\ i \neq r}}^{m} y_{ij} b^{(i)}\right).$$

But

$$a^{(k)} = By^{(k)} = \sum_{\substack{i=1,\\ i \neq r}}^{m} y_{ik} b^{(i)} + y_{rk} b^{(r)},$$

i.e.

$$a^{(k)} = \sum_{\substack{i=1,\\ i \neq r}}^{m} y_{ik} b^{(i)} + y_{rk} \frac{1}{y_{rj}} \left(a^{(j)} - \sum_{\substack{i=1,\\ i \neq r}}^{m} y_{ij} b^{(i)} \right),$$

i.e.

$$a^{(k)} = \sum_{\substack{i=1,\\ i\neq r}}^{m} \left(y_{ik} - \frac{y_{rk}y_{ij}}{y_{rj}}\right) b^{(i)} + \frac{y_{rk}}{y_{rj}} a^{(j)},$$

$$= \sum_{\substack{i=1,\\ i\neq r}}^{m} \hat{y}_{ik} b^{(i)} + \hat{y}_{rk} a^{(j)} = \hat{B}\hat{y}^{(k)};$$

which proves relation (iii).

Next, let us consider the expression for $(\hat{z}_k - c_k)$. By definition

$$(\hat{z}_k - c_k) = \sum_{i=1}^{m} \hat{c}_{Bi}\hat{y}_{ik} - c_k$$

$$= \sum_{\substack{i=1,\\ i\neq r}}^{m} c_{Bi}\left(y_{ik} - \frac{y_{rk}y_{ij}}{y_{rj}}\right) + c_j\frac{y_{rk}}{y_{rj}} - c_k$$

$$= \sum_{i=1}^{m} c_{Bi}\left(y_{ik} - \frac{y_{rk}y_{ij}}{y_{rj}}\right) + c_j\frac{y_{rk}}{y_{rj}} - c_k$$

$$= \left(\sum_{i=1}^{m} c_{Bi}y_{ik} - c_k\right) - \frac{y_{rk}}{y_{rj}}\left(\sum_{i=1}^{m} c_{Bi}y_{ij} - c_j\right),$$

i.e. $(\hat{z}_k - c_k) = (z_k - c_k) - \frac{y_{rk}}{y_{rj}}(z_j - c_j)$, which proves result (iv). □

3.5 A Useful Observation

Theorems 3.4.2 and 3.4.6 could have also been proved in an alternate way. This is because the current basis matrix $B = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, b^{(r)}, b^{(r+1)}, \ldots, b^{(m)}]$ and the next basis matrix $\hat{B} = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, a^{(j)}, b^{(r+1)}, \ldots, b^{(m)}]$ differs only by one column. Since in the current tableau the inverse of the current basis matrix, namely B^{-1}, is known, we could possibly use this to get $\hat{B}^{-1}$. If this updation of the inverse of the basis matrix could be done in a simple and efficient manner then we could possibly implement the simplex method in a different manner where rather than having $y^{(j)}$ columns in the tableau, we have B^{-1} in the tableau. This will not only reduce the size of the tableau to be handelled but also allow LPP's of bigger size to be solved because the data (c, b *and* A) could be stored separately. Such an implementation of the simplex method is available in the literature and is called the *revised simplex method.*

We shall discuss the revised simplex method in the next section where a relationship between B^{-1} and $(\hat{B})^{-1}$ will be utilized.

Relationship between B^{-1} and $(\hat{B})^{-1}$.

Let, as before $B = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, b^{(r)}, b^{(r+1)}, \ldots, b^{(m)}]$, $y^{(j)} = B^{-1}a^{(j)}$ and $\hat{B} = [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, a^{(j)}, b^{(r+1)}, \ldots, b^{(m)}]$. Also let $F = (e_1, e_2, \ldots, e_{r-1}, y^{(j)}, e_{r+1}, \ldots, e_m)$ be an $(m \times m)$ matrix, where $e_1 = col(1, 0, 0, \ldots, 0)$, $e_2 = col(0, 1, 0, \ldots, 0), \ldots, e_m = col(0, 0, 0, \ldots, 1)$ and $y^{(j)} = col(y_{1j}, y_{2j}, \ldots, y_{mj})$. Then

$$\begin{aligned} BF &= [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, By^{(j)}, b^{(r+1)}, \ldots, b^{(m)}] \\ &= [b^{(1)}, b^{(2)}, \ldots, b^{(r-1)}, a^{(j)}, b^{(r+1)}, \ldots, b^{(m)}] \\ &= \hat{B}. \end{aligned}$$

Hence

$$(\hat{B})^{-1} = (BF)^{-1} = F^{-1}B^{-1} = EB^{-1}. \tag{3.32}$$

But the matrix $E(= F^{-1})$ can be computed very easily provided $y_{rj} \neq 0$. It can be verified that

$$E = [e_1, e_2, \ldots, e_{r-1}, \xi, e_{r+1}, \ldots, e_m],$$

where

$$\xi = col\left(\frac{-y_{1j}}{y_{rj}}, \frac{-y_{2j}}{y_{rj}}, \ldots, \frac{-y_{r-1j}}{y_{rj}}, \frac{1}{y_{rj}}, \frac{-y_{r+1j}}{y_{rj}}, \ldots, \frac{-y_{mj}}{y_{rj}}\right).$$

So E is 'almost' an identity matrix except that the rth column is the vector ξ. Once $(\hat{B})^{-1}$ is known, we can compute all entries in the tableau easily. For example

$$\begin{aligned} \hat{x}_B &= (\hat{B})^{-1}b \\ &= EB^{-1}b = E(B^{-1}b) = Ex_B \\ &= [e_1, e_2, \ldots, e_{r-1}, \xi, e_{r+1}, \ldots, e_m]x_B, \end{aligned}$$

which, as before gives

$$\hat{x}_{B_i} = \begin{cases} x_{B_i} - \dfrac{x_{B_r}y_{ij}}{y_{rj}}, & (i \neq r) \\ \dfrac{x_{B_r}}{y_{rj}}, & (i = r). \end{cases}$$

Of course $\hat{x}_B$ so far is only a basic solution because we have only enforced the condition $y_{rj} \neq 0$. To make $\hat{x}_B$ a b.f.s and that too an improved one, we have to again put conditions on r and j as in Theorem 3.4.2 which will give the usual rules for a column to enter and column to leave.

Example 3.5.1 *Noting that the matrices* $B = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix}$ *and* $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *differ only by one column, obtain* B^{-1}.

Solution We have to use the relation (3.32) which in the context of the example becomes $B^{-1} = EI$, where $E = (e_1, \xi) = \begin{pmatrix} 1 & \vdots \\ & \vdots \xi \\ 0 & \vdots \end{pmatrix}$ as B and I differ only by one column at the position of second column. Next we have to find the vector ξ. For this we need to compute the vector $y^{(2)}$ for the given $a^{(2)} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$. This gives

$$y^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

As B and I differ at the position of second column, in terms of our notation $r = 2$. Therefore $\xi = col(-2/4, 1/4) = col(-1/2, 1/4)$, which gives

$$B^{-1} = \begin{bmatrix} 1 & -1/2 \\ 0 & 1/4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1/2 \\ 0 & 1/4 \end{bmatrix}.$$

3.6 The Revised Simplex Method

In the last section, while deriving the updating formulae $(\hat{B})^{-1} = EB^{-1}$, we noted that we can implement the simplex method somewhat differently where we store the elements of current B^{-1} (and not the $y^{(j)}$ columns) in the tableau. Such an implementation of the simplex method is called the *revised simplex method*. So theoretically, the revised simplex method and the simplex method are exactly same -it is only the tableau which is different here and is of much smaller size because B^{-1} is a matrix of order $(m \times m)$ only.

The main advantage of the revised simplex method is that here we handle a tableau of much smaller size. Also this is useful in solving large scale LPP's (where n is very large) because here the data (c, b, and A) is stored separately and therefore does not change during the implementation. In contrast, for the simplex method, the initial data is stored in the tableau which itself changes during the implementation because we store the updated $y^{(j)}$ columns in the tableau. Here we may note that the simplex method requires all $y^{(j)}$ columns to be evaluated first so that $(z_j - c_j)$ can be computed. But in the revised simplex method, as current B^{-1} is known, we first compute all $(z_j - c_j)$ (and this will not require knowledge of $y^{(j)}$) and then compute *only* that $y^{(j)}$ which is needed, i.e. the one which corresponds to the negative most value of $(z_j - c_j)$. Since

the updation formulae $(\hat{B})^{-1} = EB^{-1}$ is simple and efficient, the revised simplex method implementation is certainly attractive.

The revised simplex method has another advantage over the simplex method. In most of economic applications we need to know the *shadow prices* (dual variables), which (as we shall see in the sequel) are available in the revised simplex tableau itself and no additional computational effort is needed to get them. Also certain efficient decomposition, schemes for solving structured (but large) LPP's do employ the revised simplex method for their implementation.

Here we discuss the implementation of the revised simplex method for the case when artificial variables are not required and m slack variables themselves give the $(m \times m)$ identity basis matrix to start with. This case we term as *standard form I*. The case of *standard form II*, where some artificial variables are required is illustrated by an example only.

Revised Simplex Method For Standard Form I

In this form, the m slack variables themselves give the $(m \times m)$ identity basis matrix to start with and hence the given LPP has the following form

$$
\begin{aligned}
&\textit{Max} && z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
&\text{subject to} \\
& && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\
& && \qquad \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\
& && x_1 \geq 0, \ldots, x_n \geq 0\,,
\end{aligned}
\tag{3.33}
$$

where $b_i \geq 0\,(i = 1, \ldots, m)$.

Now, in contrast to the simplex method, in the revised simplex method we make the objective function also as a part of the constraints so that problem (3.33) can be written as

$$\begin{aligned} &Max \quad z \\ &\text{subject to} \end{aligned} \tag{3.34}$$

$$\begin{aligned}
z-(c_1x_1+c_2x_2+\ldots+c_nx_n)-0x_{n+1}-0x_{n+2}+\ldots+0x_{n+m} &= 0\\
a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n+\ x_{n+1} &= b_1\\
a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n+\ 0x_{n+1}+x_{n+2} &= b_2\\
\vdots &\\
a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n+\ 0x_{n+1}+\ldots+x_{n+m} &= b_m\\
x_1,\ldots,x_{n+m} &\geq 0.
\end{aligned}$$

Here z represents the objective function and therefore can have any sign. Our aim is to find a solution of (3.34) where z is as large as possible. Now the m constraints of (3.34) constitute a system of $(m+1)$ linear equations in $(n+m+1)$ unknowns and this is expressed as

$$\left[\begin{array}{c|cccccccc}
1 & -c_1 & -c_2 & \ldots & -c_n & 0 & 0 & \ldots & 0\\
\hline
0 & a_{11} & a_{12} & \ldots & a_{1n} & 1 & 0 & \ldots & 0\\
0 & a_{21} & a_{22} & \ldots & a_{2n} & 0 & 1 & \ldots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & a_{m1} & a_{m2} & \ldots & a_{mn} & 0 & 0 & \ldots & 1
\end{array}\right]\begin{bmatrix} z\\ x_1\\ x_2\\ \vdots\\ x_{n+m}\end{bmatrix} = \begin{bmatrix} 0\\ b_1\\ b_2\\ \vdots\\ b_m\end{bmatrix}, \tag{3.35}$$

i.e.

$$\begin{bmatrix} 1 & -c^T\\ 0 & A\end{bmatrix}\begin{bmatrix} z\\ x\end{bmatrix} = \begin{bmatrix} 0\\ b\end{bmatrix} \tag{3.36}$$

where $c^T = (-c_1, -c_2, \ldots, -c_n, 0, \ldots, 0)$, and

$$A = \begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1n} & 1 & 0 & \ldots & 0\\
a_{21} & a_{22} & \ldots & a_{2n} & 0 & 1 & \ldots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
a_{m1} & a_{m2} & \ldots & a_{mn} & 0 & 0 & \ldots & 1
\end{bmatrix}.$$

Let us define $a_R^{(j)} = col(-c_j, a_{1j}, \ldots, a_{mj}) = \begin{bmatrix} -c_j\\ a^{(j)}\end{bmatrix}$ $(j = 1, \ldots, n+m)$ and $b^{(R)} = col(0, b_1, \ldots, b_m) = \begin{bmatrix} 0\\ b\end{bmatrix}$, where the subscript/superscript R refers to the revised simplex method. Now in (3.35), z has to be a basic variable always because it represents the objective function which we wish to maximize and therefore the first column of the basic matrix B_R for (3.35) is always an identity column. The remaining m columns of B_R (we

note that B_R is an $(m+1)\times(m+1)$ basis matrix for the system (3.35)) have to come from columns $a_R^{(j)}$ depending upon which m columns of A currently constitute the basis matrix B for the system $Ax = b$. Hence B_R has the following form

$$B_R = \left[\begin{array}{c|c} 1 & -c_{B1} - c_{B2} \ldots - c_{Bm} \\ \hline 0 & \\ \vdots & B \\ 0 & \end{array}\right]$$

i.e.

$$B_R = \begin{bmatrix} 1 & -c_B^T \\ 0 & B \end{bmatrix}. \tag{3.37}$$

Now using the partition method of finding inverse, we get

$$B_R^{-1} = \begin{bmatrix} 1 & c_B^T B^{-1} \\ 0 & B^{-1} \end{bmatrix}, \tag{3.38}$$

where $(m+1)$ columns of B_R^{-1} are denoted as $e_1, \beta_1, \beta_2, \ldots, \beta_m$, e_1 being the identity column, i.e. $e_1 = col(1, 0, \ldots, 0)$. Thus $B_R^{-1} = [e_1, \beta_1, \beta_2, \ldots, \beta_m]$. Now in analogy with the simplex method we compute

$$\begin{aligned} y_R^{(j)} &= B_R^{-1} a_R^{(j)} \\ &= \begin{bmatrix} 1 & c_B^T B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} -c_j \\ a^{(j)} \end{bmatrix} \\ &= \begin{bmatrix} -c_j + c_B^T B^{-1} a^{(j)} \\ B^{-1} a^{(j)} \end{bmatrix} \\ &= \begin{bmatrix} -c_j + c_B^T y^{(j)} \\ y^{(j)} \end{bmatrix} \\ &= \begin{bmatrix} (z_j - c_j) \\ y^{(j)} \end{bmatrix}. \end{aligned} \tag{3.39}$$

By equation (3.39) we observe that the first component of $y_R^{(j)}$ gives the value of $(z_j - c_j)$ and remaining m components give the usual $y^{(j)}$ column which is computed in the simplex method. The main point to be noted here is that the value of $(z_j - c_j)$ is computed in a way which does not need $y^{(j)}$, where as in the simplex method we first compute $y^{(j)}$ and then use them to compute $(z_j - c_j)$.

In practice to compute $(z_j - c_j)$, equation (3.39) tells us that we have to take the first row of the matrix B_R^{-1} and then take its dot product with the vector $a_R^{(j)}$, i.e.

$$(z_j - c_j) = (\text{First row of } B_R^{-1}).(\text{column } a_R^{(j)}). \tag{3.40}$$

Also

$$x_B^{(R)} = B_R^{-1} b^{(R)} = \begin{bmatrix} 1 & c_B^T B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} z(x_B) \\ x_B \end{bmatrix}; \tag{3.41}$$

i.e. the first component of the vector $x_B^{(R)}$ gives the current value of the objective function and the remaining m components of $x_B^{(R)}$ give the current b.f.s x_B.

As for the standard form-I, initially $B = I$ so inverse of the initial basis matrix for the revised simplex method is

$$B_R^{-1} = \left[\begin{array}{c|cccc} 1 & 0 & 0 & \dots & 0 \\ \hline 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{array}\right].$$

Also the initial revised simplex tableau looks like

Variable in the basis	e_1 (z)	β_1 (x_{n+1})	β_2 (x_{n+2})	$\dots$ $\dots$	β_m (x_{n+m})	$x_B^{(R)}$
z	1	0	0	$\dots$	0	0
x_{n+1}	0	1	0		0	b_1
x_{n+2}	0	0	1		0	b_2
$\vdots$	$\vdots$		$\vdots$			$\vdots$
x_{n+1}	0	0	0	0	0	b_m

The data $a_R^{(j)}$ $(j = 1, \dots, n)$ and $b^{(R)}$ is stored separately and access to a particular column is made as and when it is needed. Now we compute $(z_j - c_j)$ for nonbasic columns $a_R^{(j)}$ as per equation (3.40) and find the negative most value of $(z_j - c_j)$, say $(z_k - c_k)$. So we next compute $y^{(k)}$ only by computing the full vector $y_R^{(k)}$, no other $y_R^{(j)}$ is computed as no other $y^{(j)}$ is needed. Once $y^{(k)}$ is known, we find a variable to leave the basis as per the usual minimum ratio criteria. Now we need to update the current basis inverse (B_R^{-1}) to the new basis inverse, namely $(\hat{B_R})^{-1}$. For this we use the relationship

$$(\hat{B_R})^{-1} = EB_R^{-1}$$

as derived in the last section, where

$$E = [e_1, e_2, \dots, e_{r-1}, \xi, e_{r+1}, \dots, e_m],$$

and

$$\xi = col\left(\frac{-y_{1k}}{y_{rk}}, \frac{-y_{2k}}{y_{rk}}, \dots, \frac{-y_{r-1k}}{y_{rk}}, \frac{1}{y_{rk}}, \frac{-y_{r+1k}}{y_{rk}}, \dots, \frac{-y_{mk}}{y_{rk}}\right).$$

Once $(\hat{B}_R)^{-1}$ is known the new revised simplex tableau is known and then we continue till all $(z_j - c_j) \geq 0$ or there is an indication of unbounded solution.

We now illustrate the working of the revised simplex method.

Example 3.6.1 *Solve the following LPP by the revised simplex method*

$$\begin{aligned} &Max && z = 2x_1 + x_2 \\ &subject\ to \\ &&& 3x_1 + 4x_2 \leq 6 \\ &&& 6x_1 + x_2 \leq 3 \\ &&& x_1, x_2 \geq 0. \end{aligned} \tag{3.42}$$

Solution Problem (3.42) is clearly in the standard form-I because the slack variables x_3 and x_4 will give a (2×2) identity matrix to startwith. So we rewrite the problem as

$$\begin{aligned} &\text{Max} && z \\ &\text{subject to} \\ &&& z - 2x_1 - x_2 - 0x_3 - 0x_4 = 0 \\ &&& 3x_1 + 4x_2 + x_3 = 6 \\ &&& 6x_1 + x_2 + x_4 = 3 \\ &&& x_1, x_2, x_3, x_4 \geq 0. \end{aligned} \tag{3.43}$$

Now as explained, the initial revised simplex tableau for problem (3.43) is

	Variable in the basis	e_1 (z)	β_1 (x_3)	β_2 (x_4)	$x_B^{(R)}$	$y_R^{(1)}$
	z	1	0	0	0	−2
	x_3	0	1	0	6	3
←	x_4	0	0	1	3	6
						↑

First iteration

Step 1 Given the initial simplex tableau, we first compute $(z_j - c_j)$ for nonbasic columns and if all of these are nonnegative then the current solution $x_B^{(1)}$ is optimal, otherwise we find the negative most $(z_j - c_j)$ to identify $(z_k - c_k)$. For our example

$$\begin{aligned} (z_1 - c_1) &= (\text{first row of } B_R^{-1})\, a_R^{(1)} \\ &= (1,0,0)\begin{pmatrix} -2 \\ 3 \\ 6 \end{pmatrix} = -2 \end{aligned}$$

$$(z_2 - c_2) = (\text{first row of } B_R^{-1})\, a_R^{(2)}$$
$$= (1,0,0)\begin{pmatrix} -1 \\ 4 \\ -1 \end{pmatrix} = -1,$$

giving the negative most value as -2 for $k = 1$. Thus the variable x_1 becomes a basic variable.

Step 2 Once negative most value $(z_k - c_k)$ is identified, we compute the vector $y_R^{(k)}$ only, by using the relation $y_R^{(k)} = B^{-1}a_R^{(k)}$. For our example $k = 1$ and so we compute $y_R^{(1)}$ only to get

$$y_R^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -2 \\ 3 \\ 6 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \\ 6 \end{bmatrix},$$

which we augment with the current tableau as shown.

Step 3 Next we find a variable to leave the basis by employing the usual minimum ratio criteria. In this context we note that z will never be considered to leave the basis as it represents the objective function which we wish to maximize.

In our example we evaluate min $(6/3, 3/6)$ which corresponds to the variable x_4 and therefore the variable x_4 leaves the basis.

Step 4 Now we update the current inverse B_R^{-1} to get the new inverse $(\hat{B}_R)^{-1}$ which is given by EB_R^{-1}. In our example, as x_1 is becoming a basic variable and x_4 is becoming nonbasic, we have

$$E = [e_1, e_2, \xi],$$

where

$$\xi = col(1/3, -1/2, 1/6).$$

Thus

$$E = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}$$

and

$$(\hat{B}_R)^{-1} = EB_R^{-1} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}.$$

Also

$$\hat{x}_B^{(R)} = (\hat{B}_R)^{-1}b^{(R)} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}\begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 9/2 \\ 1/2 \end{bmatrix}.$$

We shall follow the convention that current basis inverse is B_R^{-1} and the next basis inverse is $(\hat{B}_R)^{-1}$ irrespective of what is the number of current iteration. Also we shall always write $x_B^{(R)}$ no matter at what iteration we are

	Variable in the basis	e_1 (z)	β_1 (x_3)	β_2 (x_1)	$x_B^{(R)}$	$y_R^{(2)}$
	z	1	0	1/3	1	−2/3
←	x_3	0	1	−1/2	9/2	7/2
	x_1	0	0	1/6	1/2	1/6
						↑

Second iteration

Step 1 We have $(z_4 - c_4) = (1,0,1/3)\begin{pmatrix}0\\0\\1\end{pmatrix} = 1/3,$

$$(z_2 - c_2) = (1,0,1/3)\begin{pmatrix}-1\\4\\1\end{pmatrix} = -2/3,$$

therefore x_2 becomes a basic variable.

Step 2 We find $y_R^{(2)} = \begin{bmatrix}1 & 0 & 1/3\\0 & 1 & -1/2\\0 & 0 & 1/6\end{bmatrix}\begin{bmatrix}-1\\4\\1\end{bmatrix} = \begin{bmatrix}-2/3\\7/2\\1/6\end{bmatrix}.$

Step 3 As Min $(9/2/7/2, 1/2/1/6) = Min(9/7, 3)$ occurs for the variable x_3 it becomes a nonbasic variable.

Step 4 Now $E = [e_1, \xi, e_3]$ where

$$\xi = col\left(\frac{-(-2/3)}{7/2}, \frac{2}{7}, \frac{(-1/6)}{7/2}\right) = col(4/21, 2/7, -1/21),$$

$$(\hat{B}_R)^{-1} = EB_R^{-1} = \begin{bmatrix}1 & 4/21 & 0\\0 & 2/7 & 0\\0 & -1/21 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 1/3\\0 & 1 & -1/2\\0 & 0 & 1/6\end{bmatrix} = \begin{bmatrix}1 & 4/21 & 5/21\\0 & 2/7 & -1/7\\0 & -1/21 & 4/21\end{bmatrix}$$

$$\hat{x}_B^{(R)} = (\hat{B}_R)^{-1}b^{(R)} = \begin{bmatrix}1 & 4/21 & 5/21\\0 & 2/7 & -1/7\\0 & -1/21 & 4/21\end{bmatrix}\begin{bmatrix}0\\6\\3\end{bmatrix} = \begin{bmatrix}13/7\\9/7\\2/7\end{bmatrix}.$$

Therefore the next revised simplex tableau is

Variable in the basis	e_1 (z)	β_1 (x_2)	β_2 (x_1)	$x_B^{(R)}$
z	1	4/21	5/21	13/7
x_2	0	2/7	−1/7	9/7
x_1	0	−1/21	4/21	2/7

Third iteration

Step 1 We have $(z_3 - c_3) = (1, 4/21, 5/21)\begin{pmatrix}0\\1\\0\end{pmatrix} = 4/21$ and

$$(z_4 - c_4) = (1, 4/21, 5/21)\begin{pmatrix}0\\0\\1\end{pmatrix} = 5/21.$$

As both of these are non-negative, the current solution is optimal. Therefore, the optimal value $z^\star$ of the given LPP is 13/7 and the optimal solution is $(x_1^\star = 9/7, x_2^\star = 2/7)$.

The Revised Simplex Method For The Standard Form-II

We next consider the standard form-II, where the artificial variables are required. There could be many versions for the implementation of the revised simplex method for this case but the easiest and most natural seems to be to use the Big-M method and solve the same by the revised simplex method. We illustrate this by the following example

$$\begin{aligned} Max \quad & z = 4x_1 + 3x_2 \\ \text{subject to} \quad & \\ & x_1 + x_2 \le 8 \\ & 2x_1 + x_2 \ge 10 \\ & x_1, x_2 \ge 0. \end{aligned} \tag{3.44}$$

Let problem (3.44) be solved by the Big-M method but rather than employing the simplex method we wish to employ the revised simplex method. We have

$$\begin{aligned} Max \quad & z = 4x_1 + 3x_2 + 0x_3 + 0x_4 - Mx_{a1} \\ \text{subject to} \quad & \\ & x_1 + x_2 + x_3 = 8 \\ & 2x_1 + x_2 - x_4 + x_{a1} = 10 \\ & x_1, x_2, x_3, x_4, x_{a1} \ge 0 \end{aligned} \tag{3.45}$$

which in the revised simplex format is expressed as

$$\begin{aligned} Max \quad & z \\ \text{subject to} \quad & \\ & z - 4x_1 - 3x_2 - 0x_3 - 0x_4 + Mx_{a1} = 0 \\ & x_1 + x_2 + x_3 = 8 \\ & 2x_1 + x_2 - x_4 + x_{a1} = 10 \\ & z \text{ is unrestricted}, x_1, x_2, x_3, x_4, x_{a1} \ge 0. \end{aligned} \tag{3.46}$$

Here $c_B^T = (0, -M)$ and is $c_B^T B^{-1} = (0, -M)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = (0, -M)$.

Therefore the initial revised simplex tableau for problem (3.46) is

	Variable in the basis	e_1 (z)	$\beta_1^{(1)}$ (x_3)	$\beta_2^{(1)}$ (x_{a1})	$x_B^{(R)}$	$y_R^{(1)}$
	z	1	0	$-M$	$-10M$	$-4-2M$
	x_3	0	1	0	8	1
←	x_{a1}	0	0	1	10	$\boxed{2}$
						↑

because

$$x_B^{(R)} = B_R^{-1} b^{(R)} = \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} -10M \\ 8 \\ 10 \end{bmatrix}.$$

First iteration

Step 1 We have $(z_1 - c_1) = (1, 0, -M)\begin{pmatrix} -4 \\ 1 \\ 2 \end{pmatrix} = -4 - 2M$

$$(z_2 - c_2) = (1, 0, -M)\begin{pmatrix} -3 \\ 1 \\ 1 \end{pmatrix} = -3 - M$$

$$(z_4 - c_4) = (1, 0, -M)\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = M.$$

So the negative most value of $(z_j - c_j)$, i.e. $(z_k - c_k) = -4 - 2M$, i.e. $k = 1$ and x_1 becomes a basic variable.

Step 2 Next we compute $y_R^{(1)}$, i.e.

$$y_R^{(1)} = \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -4 - 2M \\ 1 \\ 2 \end{bmatrix},$$

and append the same to the initial tableau as shown here.

Step 3 The min ratio, namely min(8/10, 10/2) is attained for the variable x_{a1} so x_{a1} leaves the basis as it becomes a non basic variable.

Step 4 Now $E = [e_1, e_2, \xi]$, where

where $\xi = col\left(\frac{-(-4-2M)}{2}, -1/2, 1/2\right)$,

$$(\hat{B}_R)^{-1} = EB_R^{-1} = \begin{bmatrix}1 & 0 & 2+M\\ 0 & 1 & -1/2\\ 0 & 0 & 1/2\end{bmatrix}\begin{bmatrix}1 & 0 & -M\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 2\\ 0 & 1 & -1/2\\ 0 & 0 & 1/2\end{bmatrix}$$

$$x_B^{(R)} = (\hat{B}_R)^{-1}b^{(1)} = \begin{bmatrix}1 & 0 & 2\\ 0 & 1 & -1/2\\ 0 & 0 & 1/2\end{bmatrix}\begin{bmatrix}0\\ 8\\ 10\end{bmatrix} = \begin{bmatrix}20\\ 3\\ 5\end{bmatrix}.$$

Therefore the next revised simplex tableau is

	Variable in the basis	e_1 (z)	β_1 (x_3)	β_2 (x_1)	$x_B^{(R)}$	$y_R^{(4)}$
	z	1	0	2	20	−2
←	x_3	0	1	−1/2	3	1/2
	x_1	0	0	1/2	5	−1/2

↑

Second iteration

Step 1 We have $(z_2 - c_2) = (1,0,2)\begin{pmatrix}-4\\ 1\\ 2\end{pmatrix} = 0$ and

$$(z_4 - c_4) = (1,0,2)\begin{pmatrix}0\\ 0\\ -1\end{pmatrix} = -2.$$

Therefore x_4 becomes a basic variable.

Step 2 We evaluate $y_R^{(4)} = \begin{bmatrix}1 & 0 & 2\\ 0 & 1 & -1/2\\ 0 & 0 & 1/2\end{bmatrix}\begin{bmatrix}0\\ 0\\ -1\end{bmatrix} = \begin{bmatrix}-2\\ 1/2\\ -1/2\end{bmatrix}$

Step 3 The minimum ratio occurs for the variable x_3 so x_3 becomes a nonbasic variable.

Step 4 Now $E = [e_1, \xi, e_3]$ where

$$\xi = col\left(\frac{-(-2)}{1/2}, 2, \frac{-(-1/2)}{2}\right) = col(4,2,1),$$

$$(\hat{B}_R)^{-1} = \begin{bmatrix}1 & 4 & 0\\ 0 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 2\\ 0 & 1 & -1/2\\ 0 & 0 & 1/2\end{bmatrix} = \begin{bmatrix}1 & 4 & 0\\ 0 & 2 & -1\\ 0 & 1 & 0\end{bmatrix}$$

$$x_B^{(R)} = \begin{bmatrix} 1 & 4 & 0 \\ 0 & 2 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} 32 \\ 6 \\ 8 \end{bmatrix}$$

Therefore the next revised simplex tableau is

Variable in the basis	e_1 (z)	β_1 (x_4)	β_2 (x_1)	$x_B^{(R)}$
z	1	4	0	32
x_4	0	2	−1	6
x_1	0	1	0	8

Third iteration

Step 1 We have $(z_2 - c_2) = (1,4,0)\begin{pmatrix} -4 \\ 1 \\ 0 \end{pmatrix} = 0$

$$(z_3 - c_3) = (1,4,0)\begin{pmatrix} -3 \\ 1 \\ 1 \end{pmatrix} = 1$$

$$(z_4 - c_4) = (1,4,1)\begin{pmatrix} M \\ 0 \\ 1 \end{pmatrix} = M - 1.$$

As all $(z_j - c_j) \geq 0$, the current solution namely, $(x_1^\star = 8,\ x_2^\star = 0)$ is optimal and the optimal value $z^\star = 32$.

3.7 Summary and Additional Notes

- Section 3.2 presented certain basic results on the geometry of general LPP's in $\mathbf{R}^n$ which are then used to characterize the optimal solution.
- Section 3.4 is devoted to provide the proofs of all the main theorems used in describing the simplex algorithm.
- Section 3.6 describes a different and more useful implementation of the simplex method, namely the revised simplex method.
- The revised simplex method was developed by Dantzig and Orchard-Hays in 1954 (see Dantzig [44]).
- The proofs of the main theorems in the simplex algorithm are motivated by Hadley [72] and Taha [154].
- For linear programming problems with bounded variables, Dantzig presented a modified simplex method in 1955. The same method was independently developed by Charnes and Lemeke in 1954. Though we have not discussed LPP's with bounded variables, these problems are important and for further study on this topic, Murty [117], Bazaraa et al. [12] and Taha [154] are good references.

- There are certain complexity issues with regard to the simplex algorithm which along with the Karmarkar's algorithm are discussed in Chapter 13 of this book.
- Solving large LPP's is always a problem but certain structured LPP's can be solved efficiently by employing various decomposition schemes available in the literature. The texts by Lasdon [101] and Wismer [166] are good references in this regard.
- It is natural to think the possibility of replacing more than one variable in the simplex method so as to reduce the number of iterations needed to solve the given LPP. Although there does not seem to be any major computational gain by this approach, some such works have been reported in the literature, e.g. Paranjape [126].

3.8 Exercises

3.1 *Without sketching the graph find all the vertices of the set of feasible solutions of*

$$\begin{aligned} -x_1 + x_2 &\leq 1 \\ 2x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

3.2 *Use the simplex method to find a corner point of the region given by*

$$\begin{aligned} x_1 + x_2 &\geq 1 \\ x_1 + x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

3.3 *Let* $S = \{(x_1, x_2) : x_1 - x_2 \leq 4, x_1 + x_2 \geq -3, x_2 \leq 8\}$. *Find all extreme points of* S *and hence express the point* $(2, 1)$ *as a convex combination of these extreme points.*

3.4 *Use the simplex method to verify that the following LPP has unbounded solution*

$$\begin{aligned} &\textit{Max} && x_1 + 2x_2 \\ &\textit{subject to} \\ & && -2x_1 + x_2 + x_3 \leq 2 \\ & && -x_1 + x_2 - x_3 \leq 1 \\ & && x_1,\ x_2,\ x_3 \geq 0. \end{aligned}$$

From the final simplex tableau construct a feasible solution for which the objective function is greater than 2000.

3.5 *The optimal simplex tableau of a LPP is*

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 6$	0	1	2	−1
$x_1 = 2$	1	0	−1	1
26	0	0	2	1

Given that x_3 and x_4 are slack variables and the starting basis was $(x_3, x_4)^T$, write the original LPP.

3.6 *The following is the current simplex tableau of a LPP in the maximization form*

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	y_5	y_6	y_7
$x_1 = c$	1	0	0	0	a	3	2
$x_2 = f$	0	1	0	0	2	d	−1
$x_3 = 1$	0	0	1	0	−1	0	g
$x_4 = 3$	0	0	0	1	4	−1	6
$z = 5$	0	0	0	0	3	−4	1

Determine the conditions on a,c,d,f,g so that the current tableau and the updated tableau represent respectively

1. *Non-degenerate and degenerate b.f.s's.*
2. *Non-degenerate and non-degenerate b.f.s's.*
3. *Degenerate and non-degenerate b.f.s's.*
4. *Degenerate and degenerate b.f.s's.*

3.7 *Solve the following linear programming problem by the simplex method. Also, at each iteration identify B and B^{-1}.*

$$\begin{aligned} \textit{Max} \quad & 3x_1 + 2x_2 + x_3 \\ \textit{subject to} \quad & \\ & 2x_1 - 3x_2 + 2x_3 \leq 3 \\ & -x_1 + x_2 + x_3 \leq 5 \\ & x_1, x_2, x_3 \geq 0 . \end{aligned}$$

3.8 *Solve the following problem by the simplex method starting with the corner point (4,2) and show the movement graphically*

$$\begin{aligned} \textit{Max} \quad & x_1 + 3x_2 \\ \textit{subject to} \quad & \\ & -x_1 + x_2 \leq 2 \\ & x_1 + 2x_2 \leq 8 \\ & 2x_1 - x_2 \leq 6 \\ & x_1, x_2 \geq 0 . \end{aligned}$$

3.9 *Use the simplex method to show that the following LPP has unbounded solution*

$$\begin{aligned} &Max && 4x_1 + 3x_2 \\ &\text{subject to,} \\ & && x_1 + x_2 \geq 8 \\ & && 2x_1 + x_2 \geq 10 \\ & && x_1,\ x_2 \geq 0. \end{aligned}$$

Hence obtain a feasible solution for which the given objective function takes the value 300.

3.10 *Solve the following problem by the simplex method starting with the b.f.s corresponding to the corner point* $(x_1, x_2) = (4, 0)$

$$\begin{aligned} &Max && -x_1 + 2x_2 \\ &\text{subject to} \\ & && 3x_1 + 4x_2 = 12 \\ & && 2x_1 - x_2 \leq 12 \\ & && x_1, x_2 \geq 0 \end{aligned}$$

3.11 *Let following be the simplex tableau of a LPP at some iteration*

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = a$	0	c	1	1
$x_1 = b$	1	-1	0	2
	0	d	0	e

State all possible values of a, b, c, d *and* e *in each of the following so that the given statement is true*

1. *the current solution is optimal*
2. *the given LPP has unbounded solution*
3. *the current solution is not optimal but the objective function value can be increased by replacing* x_3 *by* x_2.
4. *the current solution is optimal but the LPP has many optimal solutions*

3.12 *Let it be known that 'The system of linear equations* $Ax = b$ *is consistent if and only if Rank* A = *Rank*$(A : b)$.' *Hence or otherwise, use the simplex algorithm to check that the matrices*

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 1 & 4 \\ 2 & 1 & 2 \end{pmatrix}$$

have the same rank.

3.13 *Use the simplex method to find all optimal solutions of the LPP*

$$\begin{aligned} &\textit{Min} && -2x_1 - 4x_2 \\ &\textit{subject to} \\ & && x_1 + 2x_2 \leq 4 \\ & && x_1 - x_2 \geq -1 \\ & && x_1, x_2 \geq 0 \end{aligned}$$

Verify your answer graphically.

3.14 *Use the simplex method to show that the system of linear equations*

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + x_2 &= 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

is consistent and hence obtain a solution of the given system.

3.15 *Two consecutive simplex tableaus of a given LPP are*

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
$x_4 = 6$	b	c	d	1	0
$x_5 = 1$	-1	2	e	0	1
0	a	-1	3	0	0

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
$x_1 = f$	g	2/3	2/3	1/3	0
$x_5 = 3$	h	1	$-1/3$	1/3	1
2	0	$-1/3$	j	k	0

Find the value of a, b, c, d, e, f, g, h, i, j and k.

3.16 *Use the simplex method to verify that the following LPP has unbounded solution*

$$\begin{aligned} &\textit{Max} && 2x_1 + 3x_2 \\ &\textit{subject to} \\ & && x_1 - x_2 + x_3 \leq 2 \\ & && -3x_1 + x_2 \leq 4 \\ & && x_1,\ x_2 \geq 0. \end{aligned}$$

Hence find a feasible solution for which the value of the objective function is greater than 248.

3.17 *In a simplex tableau if* $(z_j - c_j) = 10$ *for a non basic variable* x_j. *Find the change in the objective function value if* x_j *enters the basis and it is given that the minimum ratio is 2 in the pivot.*

3.18 *Use the simplex algorithm to find a solution of the system*

$$\begin{aligned} x_1 + x_2 &= 2 \\ 2x_1 + x_2 &= 12\,. \end{aligned}$$

Does the system have only one solution? How does it get checked by the simplex method.

3.19 *Consider the optimization problem*

$$\begin{aligned} &\textit{Max} && \textstyle\sum_{j=1}^{n} |c_j x_j + \alpha_j| \\ &\textit{subject to} \\ & && \textstyle\sum_{j=1}^{n} a_{ij} x_j \le b^{(i)} \quad (i = 1, 2, \ldots, m) \end{aligned}$$

1. *Is the above a LPP? Give reasons.*
2. *Can this be transformed into a LPP so as to solve the same by the simplex method. Verify your answer for the problem*

$$\begin{aligned} &\textit{Max} && 5|x_1| + 6|x_2| \\ &\textit{subject to} \\ & && 3x_1 + 4x_2 \le 6 \\ & && x_1 + 3x_2 \ge 2. \end{aligned}$$

3.20 *Let it be given that* $\begin{bmatrix} 3 & -5 & -1 \\ 2 & -1 & 0 \\ 1 & 2 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 2/5 & 1/5 \\ 0 & -1/5 & 2/5 \\ -1 & 11/5 & -7/5 \end{bmatrix}$

Use the information to determine $\begin{bmatrix} 3 & -5 & -1 \\ 2 & -1 & 0 \\ 1 & 2 & 1 \end{bmatrix}^{-1}$.

3.21 *Let* $S = \{(x_1, x_2) : |x_1| + |x_2| \le 1\}$ *and* $S_1 = S \cup \{(1,1)\}$. *Let* $Conv(S_1)$ *denote the convex hull of the set* S_1. *Use simplex algorithm to maximize* $4x_1 + 3x_2$ *over* $Conv(S_1)$.

3.22 *Let it be known that*

$$\begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1/2 & 0 \\ -1 & 1 \end{bmatrix}$$

Use this information to evaluate P^{-1} *and* Q^{-1} *where*

$$P = \begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}, \quad \text{and} \quad Q = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix}.$$

3.23 *Solve the following LPP by the revised simplex algorithm*

$$\begin{aligned} &\textit{Max} && 4x_1 + 3x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 \le 8 \\ & && 2x_1 + x_2 \le 10 \\ & && x_1, x_2 \ge 0 . \end{aligned}$$

3.24 *Let* $S = \{(-1,0),(0,1),(1,0)\}$. *Use the revised simplex algorithm to minimize* $4x_1 + 3x_2$ *over the convex hull of* S.

3.25 *Use the revised simplex algorithm to solve the following LPP*

$$\begin{aligned} &\textit{Max} && 2x_1 + 4x_2 \\ &\textit{subject to} \\ & && 2x_1 + 3x_2 \le 6 \\ & && x_1 + x_2 \ge 1 \\ & && x_1, x_2 \ge 0 . \end{aligned}$$

3.26 *Are the following statements true? Give reasons for your answer*

1. *The system of equations* $x_1 + x_2 + x_3 = 3$, $x_1 - x_2 - x_4 = 0$, $x_1, x_2, x_3 \ge 0$, *has three degenerate b.f.s.*
2. *For the LPP 'Max* $4x_1 + 3x_2 - 5x_3$, *subject to* $4x_1 + x_2 + 6x_3 \le 12, x_1, x_2 \ge 0$ x_3 *unrestricted in sign', the optimal solution* (x_1^*, x_2^*, x_3^*) *can never have* $x_1^* > 0$ *and* $x_2^* > 0$.
3. *The set* $S = \{(x_1, x_2) : x_1 \ge 0, x_2 \ge 0, x_1 + x_2 \ge 1\}$ *is a polyhedron but not a polytope.*
4. *The set* $S = \{(x_1, x_2) : x_1 \ge 1$ *or* $x_2 \ge 1\}$ *can never be the feasible region of a LPP.*
5. *For the constraints* $x_1 - x_2 + x_3 = 2$, x_1, x_2 $x_3 \ge 0$, *the point* $(x_1^* = 4, x_2^* = 2, x_3^* = 0)$ *is a b.f.s.*
6. *Let* $f : [a,b] \to \mathbf{R}$ *be given by* $f(x) = mx + c$. *If* f *attains its minimum at a point* $x^* \in (a,b)$ *then* $m = 0$.
7. *The problem 'Max* $z = 4x_1 + 3x_2$ *subject to* $|x_1 - x_2| \ge 1$' *is a LPP.*
8. *The set* $S = \{(x_1, x_2) : x_1^2 + x_2^2 \le 4, x_2^2 \ge x_1\}$ *is a convex set.*

3.27 *Construct an example of each of the following (if no such example is possible then state reasons for the same)*

1. *a convex set in* $\mathbf{R}^3$ *having four corner points.*
2. *a LPP having exactly 2 optimal solutions*
3. *a closed half space in* $\mathbf{R}^2$
4. *a LPP in which more than one degenerate b.f.s. correspond to the same corner point*
5. *a simplex in* $\mathbf{R}$
6. *a nonconvex set which is the intersection of two polytopes*
7. *an optimal solution which is not a b.f.s.*
8. *a LPP having non-convex feasible region.*

4
Duality in Linear Programming

4.1 Introduction

By now we must be comfortable in using the simplex method for solving linear programming problems. One important aspect of the simplex method is that it not only solves the given LPP (called the *primal*) but also solves another closely related LPP (called the *dual*). The other LPP, namely the dual, remains hidden in the implementation of the simplex method but its solution is readily available in the optimal simplex tableau of the primal problem itself. In the study of duality in linear programming we give a well defined procedure to construct the 'hidden' LPP (namely the dual) and establish those results which bring out meaningful relationships between the given problem (primal) and its dual. These results, besides being of theoretical and computational importance, give interesting and useful economic interpretations.

4.2 The Dual Problem: Motivation Through an Example

Let us consider the LPP

$$\begin{aligned} \text{Max} \quad & 4x_1 + 3x_2 \\ \text{subject to} \quad & \\ & x_1 + x_2 \leq 8 \\ & 2x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{4.1}$$

This problem has already been solved earlier by the simplex method to get its optimal solution as ($x_1^* = 2,\ x_2^* = 6$) and its optimal value as 26. Also the initial and the final tableaus are

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 8$	1	1	1	0
$x_4 = 10$	2	1	0	1
0	−4	−3	0	0

and

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 6$	0	1	2	−1
$x_1 = 2$	1	0	−1	1
26	0	0	2	1

respectively.

Given the LPP (4.1) let us introduce another LPP as follows

$$\begin{aligned} \text{Min} \quad & 8w_1 + 10w_2 \\ \text{subject to} \quad & \\ & w_1 + 2w_2 \geq 4 \\ & w_1 + w_2 \geq 3 \\ & w_1, w_2 \geq 0. \end{aligned} \tag{4.2}$$

In this construction (which is purely adhoc at present) the problem is taken in the 'Min' form as the given problem (4.1) is in the 'Max' form. The roles of c_j ($c_1 = 4$, $c_2 = 3$) and b_i ($b_1 = 8$, $b_2 = 10$) have been interchanged and in the constraints of problem (4.2), '$\geq$' sign has been taken as in problem (4.1) the constraints are of '$\leq$' type.

As for our example, problem (4.2) has only two decision variables (w_1 and w_2), we can solve it graphically (see Fig. 4.1) to get its optimal solution as ($w_1^* = 2$, $w_2^* = 1$) and its optimal value as 26.

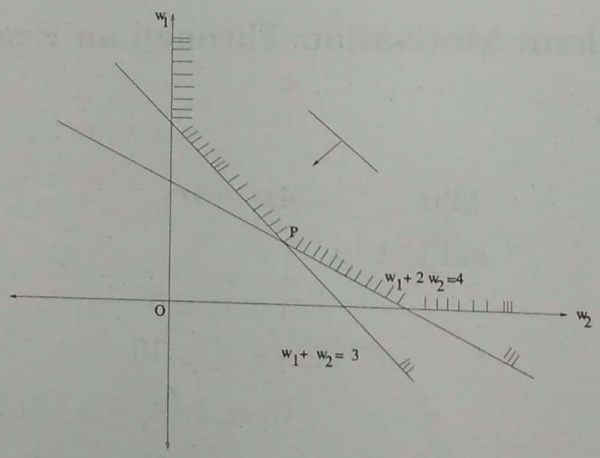


Fig. 4.1.

If we now look at the last tableau of problem (4.1) as given here, we notice two things. Firstly, both problems (4.1) and (4.2) have the same optimal value, namely, 26; and secondly, the optimal solution of problem (4.2) is really present in the last row of the optimal simplex tableau of problem (4.1). But, is it just a coincidence or is there

something deeper in it? The duality theory of linear programming asserts that this is always going to happen, i.e. if the given LPP (primal) has an optimal solution then its dual will certainly have an optimal solution and further the optimal values of the two problems will be equal.

We now proceed to the construction of the dual for a general LPP and then to establish basic duality relationship between this pair of LPP's.

4.3 Construction of the Dual

Let the given LPP (called *primal*) be

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax \leq b \\ & && x \geq 0, \end{aligned} \tag{4.3}$$

where $x \in \mathbf{R}^n$, $c \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ and A is an $(m \times n)$ real matrix.

We now construct another associated problem, called the *dual* of the primal problem (4.3), as follows

$$\begin{aligned} &\text{Min} && b^T w \\ &\text{subject to} \\ & && A^T w \geq c \\ & && w \geq 0, \end{aligned} \tag{4.4}$$

where $w \in \mathbf{R}^m$.

The LPP's (4.3)-(4.4) together are called the *primal-dual pair*. Also the components of vector x (respectively vector w) are called the *primal variables* (respectively *dual variables*) and the constraints $Ax \leq b$ (respectively $A^T w \geq c$) are called the *primal constraints* (respectively *dual constraints*). Here it may be noted that the number of dual constraints equals the number of primal variables and the number of dual variables equals the number of primal constraints. Further, if we write problem (4.4) in the 'max' form (i.e. in the form of (4.3)) and write its dual we get problem (4.3). Thus if we take the dual of dual we get back primal, i.e. dual (dual) = primal. This property of LPP is called the *symmetric duality*.

As the above construction of the primal-dual pair (4.3)-(4.4) is symmetric we call either of these two problems as primal and the other one as its dual, i.e. what we are calling as 'primal', the same may also be called as 'dual' and vice-versa. However, in our presentation we shall continue calling problem (4.3) as the primal and problem (4.4) as its dual. With this understanding, we already have an example of a primal-dual pair, namely LPP's (4.1) and (4.2).

Next let us consider the situation when the constraints of the given LPP (we sh
continue calling it primal) are of '≥' type, i.e. the given LPP has the following form

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax \geq b \\ & && x \geq 0. \end{aligned} \tag{4}$$

Now problem (4.5) can be rewritten in the form

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && (-A)x \leq (-b) \\ & && x \geq 0, \end{aligned}$$

and then using the construction (4.3)-(4.4) we can write its dual as

$$\begin{aligned} &\text{Min} && (-b)^T u \\ &\text{subject to} \\ & && (-A)^T u \geq c \\ & && u \geq 0. \end{aligned}$$

Now if we write $w = -u$, then the above problem can be written as

$$\begin{aligned} &\text{Min} && b^T w \\ &\text{subject to} \\ & && A^T w \geq c \\ & && w \leq 0. \end{aligned} \tag{4.}$$

If we now compare problems (4.4) and (4.6), then we observe that (4.6) is of the sam
form as (4.4) except here $w \leq 0$ rather than $w \geq 0$. This is because the primal constrain
are of '≥' type.

We next consider the case when the constraints of the primal are of '=' type, i.e. t
primal problem is

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax = b \\ & && x \geq 0. \end{aligned} \tag{4.}$$

Problem (4.7) can be rewritten as

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax \le b \\ & && (-A)x \le -b \\ & && x \ge 0. \end{aligned} \tag{4.8}$$

Now using the construction (4.3)-(4.4), we get the dual of above problem as

$$\begin{aligned} &\text{Min} && b^T u - b^T v \\ &\text{subject to} \\ & && A^T u - A^T v \ge c \\ & && u \ge 0 \\ & && v \ge 0. \end{aligned} \tag{4.9}$$

If we now write $w = (u - v)$ then the above problem can be written as

$$\begin{aligned} &\text{Min} && b^T w \\ &\text{subject to} \\ & && A^T w \ge c \\ & && w \text{ unrestricted in sign.} \end{aligned} \tag{4.10}$$

A comparison of (4.4) and (4.10) tells that the only difference here is that w is unrestricted in sign. Again this has happened because the primal constraints are of '=' type.

The above discussion tells that the sign of dual variables w are determined by the sign of primal constraints. If the ith primal constraint is of '$\le$' type then the ith dual variable $w_i \ge 0$; if the ith primal constraint is of '$\ge$' type then the ith dual variable $w_i \le 0$; and if the ith primal constraint is of '=' type then the ith dual variable w_i is unrestricted in sign. Here it may be noted that we have not proved exactly in this manner but above is certainly the conclusion of what we have proved in our discussion.

Just as the sign of dual variables is determined by the sign of primal constraints, because of symmetric dual nature, we expect that the sign of primal variables will determine the sign of the dual constraints. Using the construction (4.3)-(4.4), it is not difficult to prove that if $x_j \ge 0$ then the jth dual constraint will be of '$\ge$' type; if $x_j \le 0$ then the jth dual constraint will be of '$\le$' type and, if x_j unrestricted in sign, then the jth dual constraint will be of '=' type.

The above construction of the dual is valid for any general LPP (primal) which is given in the 'max' form and we summarize this construction in the form of a table called *rule table for the construction of the dual.* We read this table from 'left' to 'right' as the given LPP (primal) is in the 'max' form. But what happens if the given LPP (primal)

is in the 'min' form? The answer is obvious, we read the rule table from 'right' to 'left' and our construction will be correct because of the symmetric nature of the LPP dual.

Rule Table for the Construction of the Dual

Primal (Max)	Dual (Min)
i^{th} constraint is $\leq$ type	$w_i \geq 0$
i^{th} constraint is $\geq$ type	$w_i \leq 0$
i^{th} constraint is $=$ type	w_i unrestricted in sign
$x_j \leq 0$	j^{th} constraint is $\leq$ type
$x_j \geq 0$	j^{th} constraint is $\geq$ type
x_j unrestricted in sign	j^{th} constraint is $=$ type

We now illustrate the construction of the dual for some examples.

Example 4.3.1 *Write the dual of the following LPP*

Max $\quad -2x_1 - x_2$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 10 \\ x_1 - 2x_2 &= -8 \\ x_1 + 3x_2 &\geq 9 \\ x_1 \geq 0, \quad x_2 &\text{ unrestricted in sign.} \end{aligned} \tag{4.11}$$

Solution We observe that $x = col(x_1, x_2)$, $c = col(-2, -1)$, $b = col(10, -8, 9)$,

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -2 \\ 1 & 3 \end{pmatrix} \text{ and } A^T = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 3 \end{pmatrix}.$$

As there are three primal constraints, there will be three dual variables, i.e. $w = col(w_1, w_2, w_3)$. Also $b^T w = 10w_1 - 8w_2 + 9w_3$ and $A^T w = col(w_1 + w_2 + w_3, w_1 - 2w_2 + 3w_3)$. Now the given LPP is in the 'Max' form so we read the rule table from 'left' to 'right'. This gives the dual as

Min $\quad w = 10w_1 - 8w_2 + 9w_3$

subject to

$$\begin{aligned} w_1 + w_2 + w_3 &\geq -2 \\ w_1 - 2w_2 + 3w_3 &= -1 \\ w_1 \geq 0, w_3 \leq 0, &\ w_2 \text{ unrestricted in sign.} \end{aligned}$$

In the dual, the first constraint is of '$\geq$' type as $x_1 \geq 0$ and the second constraint is of '$=$' type as x_2 unrestricted in sign. Similarly $w_1 \geq 0$ as the first primal constraint is

of '$\leq$' type, w_2 unrestricted in sign as the second primal constraint is of '$=$' type and $w_3 \leq 0$ as the third primal constraint is of '$\geq$' type.

Example 4.3.2 *Write the dual of the following LPP*

Min $z = 2x_1 - x_2 + x_3$

subject to

$$\begin{aligned} 2x_1 + x_2 - x_3 &\leq 8 \\ -x_1 + x_3 &\geq 1 \\ x_1 + 2x_2 + 3x_3 &= 9 \\ x_1 \geq 0, x_2 \geq 0, &\ x_3 \text{ unrestricted in sign.} \end{aligned} \tag{4.12}$$

Solution As the given (primal) LPP is in the 'Min' form we read the rule table from 'right' to 'left' and get its dual as

Max $w = 8w_1 + w_2 + 9w_3$

subject to

$$\begin{aligned} 2w_1 - w_2 + w_3 &\leq 2 \\ w_1 + 2w_3 &\leq -1 \\ -w_1 + w_2 + 3w_3 &= 1 \\ w_1 \leq 0, w_2 \geq 0, &\ w_3 \text{ unrestricted in sign.} \end{aligned}$$

Remark 4.3.1 *Here it may be remarked that it is not essential to use the rule table. In fact we can always write problem (4.11) in the form of problem (4.3) and use the construction (4.4) to get its dual by the first principle. Similarly the dual of problem (4.14) can also be obtained by the first principle. We use the rule table for the convenience only and there is absolutely no compulsion to use it.*

4.4 Duality Theorems

Let us consider the primal-dual pair as

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax \leq b \\ & && x \geq 0 \end{aligned} \tag{4.13}$$

and

$$\begin{aligned} &\text{Min} && b^T w \\ &\text{subject to} \\ & && A^T w \geq c \\ & && w \geq 0. \end{aligned} \tag{4.14}$$

In this section, we present various duality results relating problems (4.13) and (4.14). In particular we prove four main theorems, namely the *weak duality theorem*, the *strong duality theorem*, the *existence theorem* and the *complementary slackness theorem*. In the following we shall write 'primal' and 'dual' problems only and it will be understood that we are referring to problems (4.13) and (4.14) respectively.

Theorem 4.4.1 (Weak Duality Theorem). *Let x be feasible for the primal and w be feasible for the dual. Then*

$$c^T x \leq b^T w.$$

Proof. Since x is feasible for the primal, we have

$$Ax \leq b, x \geq 0. \tag{4.15}$$

Similarly the feasibility of w for the dual gives

$$A^T w \geq c, w \geq 0. \tag{4.16}$$

Now $Ax \leq b$ implies $x^T A^T \leq b^T$, which on multiplication by $w(\geq 0)$ on the right gives

$$x^T A^T w \leq b^T w. \tag{4.17}$$

Similarly $A^T w \geq c$ implies $w^T A \geq c^T$, which on multiplication by $x(\geq 0)$ on the right gives

$$w^T Ax \geq c^T x. \tag{4.18}$$

But $x^T A^T w$ is a scalar and hence $x^T A^T w = (x^T A^T w)^T = w^T Ax$. Therefore (4.17) and (4.18) give $c^T x \leq b^T w$. □

Corollary 4.4.1 *Let $\overline{x}$ be feasible for the primal and $\overline{w}$ be feasible for the dual. Also let $c^T \overline{x} = b^T \overline{w}$. Then $\overline{x}$ is optimal to the primal and $\overline{w}$ is optimal to the dual.*

Proof. Let x be an arbitrary feasible point of the primal. The feasibility of $\overline{w}$ to the dual together with the weak duality theorem gives $c^T x \leq b^T \overline{w}$. But as $b^T \overline{w} = c^T \overline{x}$ is given, this gives $c^T x \leq c^T \overline{x}$, i.e. $\overline{x}$ is optimal for the primal. Similarly we can show that $\overline{w}$ is optimal to the dual. □

Remark 4.4.1 *In view of the above corollary, as soon as we get two feasible solutions, one for the primal and the other for the dual, such that the corresponding objective function values are equal, we have optimal solutions for both primal and dual problems. Thus it seems logical that an algorithm for LPP should not only maximize the value of the primal but simultaneously minimize the value of the dual as well and stop when both of these values become equal. This has been the guiding principle of most of the algorithms for solving LPP's.*

Theorem 4.4.2 (Strong Duality Theorem).

(i) Let $\bar{x}$ be an optimal solution of the primal. Then there exists a $\bar{w}$ which is optimal to the dual. Also $c^T\bar{x} = b^T\bar{w}$.

(ii) Let w^* be an optimal solution of the dual. Then there exists a x^* which is optimal to the primal. Also $b^Tw^* = c^Tx^*$.

Here part (*i*) is called the *direct duality theorem* and part (*ii*) is called the *converse duality theorem*. As dual (dual) = primal, it is enough to prove part (*i*) only.

There are many proofs of this theorem but most of these only show the existence of optimal solution $\bar{w}$ for the dual, given that $\bar{x}$ is optimal to the primal.

Our proof presented here is constructive in the sense that given $\bar{x}$, we actually construct $\bar{w}$, and hence this proof is probably more useful from an application point of view.

Proof. Without any loss of generality we can take the primal-dual pair as

$$\begin{aligned} &\text{Max} && c^Tx \\ &\text{subject to} \\ & && Ax = b \\ & && x \geq 0, \end{aligned}$$

and

$$\begin{aligned} &\text{Min} && b^Tw \\ &\text{subject to} \\ & && A^Tw \geq c \\ & && w \text{ unrestricted in sign.} \end{aligned} \tag{4.19}$$

Also, again without any loss of generality, we can assume that $\bar{x}$ which is optimal to problem (4.19) has been obtained by solving (4.19) by the simplex method. Thus $\bar{x}$ is an optimal basic feasible solution of (4.19), i.e. for some basis matrix B, $\bar{x} = (x_B = B^{-1}b,\ x_R = 0)$.

But $\bar{x}$ is optimal to problem (4.19) hence we have

$$(z_j - c_j) \geq 0 \quad (j = 1, 2, \ldots, n)$$

i.e.

$$(c_B^T y^{(j)} - c_j) \geq 0 \quad (j = 1, 2, \ldots, n)$$

i.e.

$$(c_B^T B^{-1})a^{(j)} \geq c_j \quad (j = 1, 2, \ldots, n). \tag{4.20}$$

Now if we define a vector $\bar{w} \in \mathbf{R}^m$ given by $\bar{w}^T = c_B^T B^{-1}$ then (4.20) gives $A^T\bar{w} \geq c$. This implies that $\bar{w}$ is feasible to the dual problem (4.19) as in (4.19) $\bar{w}$ is unrestricted in sign.

Thus given an optimal solution $\bar{x}$ of the primal (4.19) we have been able to construct a vector $\bar{w}$, given by $\bar{w}^T = c_B^T B^{-1}$, such that $\bar{w}$ is feasible to the dual (4.19). Now if we prove that $c^T\bar{x} = b^T\bar{w}$ then because of the weak duality theorem $\bar{w}$ will be optimal to the dual (4.19). But this is true as

$$c^T\bar{x} = c_B^T x_B = c_B^T(B^{-1}b) = (c_B^T B^{-1})b = \bar{w}^T b = b^T\bar{w}.$$

Hence the result. □

Example 4.4.1 *Write the dual of the following LPP (primal)*

$$\begin{aligned} &\textit{Max} && 4x_1 + 3x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 \le 8 \\ & && 2x_1 + x_2 \le 10 \\ & && x_1, x_2 \ge 0\,, \end{aligned} \tag{4.21}$$

and use Theorem 4.4.2 to find the solution of the dual by solving the primal.

Solution We have already seen that the dual of problem (4.21) is

$$\begin{aligned} &\text{Min} && 8w_1 + 10w_2 \\ &\text{subject to} \\ & && w_1 + 2w_2 \ge 4 \\ & && w_1 + w_2 \ge 3 \\ & && w_1, w_2 \ge 0. \end{aligned} \tag{4.22}$$

We are required to determine an optimal solution of (4.22) by solving (4.21).

From Section 4.2 we know that the initial and final simplex tableaus for problem (4.21) are

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = 8$	1	1	1	0
$x_4 = 10$	2	1	0	1
0	−4	−3	0	0

and

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_2 = 6$	0	1	2	−1
$x_1 = 2$	1	0	−1	1
26	0	0	2	1

Thus $\bar{x} = (\bar{x}_1 = 2, \bar{x}_2 = 6, \bar{x}_3 = 0, \bar{x}_4 = 0)$ and we have to find $\bar{w}$, i.e. an optimal solution of problem (4.22).

Now from the last tableau of the primal (4.21) we get $c_B = col(c_2, c_1) = col(3, 4)$, (and NOT $col(4, 3)$ because in the tableau the basic variables are $\bar{x}_2 = 6$ and $\bar{x}_1 = 2$), and $B^{-1} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$. Hence

$$\overline{w} = col(\overline{w}_1, \overline{w}_2) = c_B^T B^{-1}$$
$$= (3,4)\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

i.e. $\overline{w}_1 = 2$ and $\overline{w}_2 = 1$ is optimal to the dual (4.22). Further by the duality theorem, optimal value of the dual equals the optimal value of the primal which is 26.

Remark 4.4.2 *The components of the vector $\overline{w}$, namely $(2,1)$ are also available in the last row of the optimal simplex tableau of problem (4.21). So the obvious question is – can we just read the optimal solution of the dual directly from the last tableau of the primal so that there is no need of identifying B^{-1} and computing $c_B^T B^{-1}$? The answer is in the affirmative and that we discuss in Section 4.5.*

Now to prove the existence theorem and the complementary slackness theorem, we again consider the primal-dual pair as stated at (4.13) and (4.14).

Theorem 4.4.3 (Existence Theorem).

(i) If primal and dual both have feasible solutions then both have optimal solutions.
(ii)If primal (dual) has unbounded solution then the dual (primal) has no feasible solution.
(iii)If primal (dual) has no feasible solution but the dual (primal) has feasible solution then the dual (primal) has unbounded solution.

Proof. (i) Let x and w be feasible solutions of the primal and the dual respectively. Then by the weak duality theorem (Theorem 4.4.1) $c^T x \leq b^T w$. Since $b^T w$ is finite and is an upper bound (not necessarily the least upper bound) on the primal objective function value, we infer that the primal has an optimal solution. Similar arguments hold for the dual.

(ii) If possible let the dual has a feasible solution. Then as primal and dual both become feasible, by part (*i*), both have optimal solution. But this contradicts that primal has unbounded solution. Therefore the dual should be infeasible.

(iii) If possible let the dual has bounded (finite) optimal solution. Then by the strong duality theorem (Theorem 4.4.2) the primal should also have an optimal solution and hence certainly be feasible. But this contradicts the hypothesis that the primal is infeasible. Therefore the dual should have unbounded solution. □

The statement of the existence theorem can also be understood by the following

(Primal) \ (Dual)	feasible	infeasible
feasible	both have optimal solution	Primal has unbounded solution
infeasible	Dual has unbounded solution	This case is possible, i.e. primal and dual both could be infeasible

We now give an example to illustrate that both primal and dual could be infeasible. For this let the primal problem be

$$\begin{aligned} \text{Max} \quad & 3x_1 + 4x_2 \\ \text{subject to} \quad & \\ & x_1 - x_2 \leq -1 \\ & -x_1 + x_2 \leq 0 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Then its dual is

$$\begin{aligned} \text{Min} \quad & -w_1 \\ \text{subject to} \quad & \\ & w_1 - w_2 \geq 3 \\ & -w_1 + w_2 \geq 4 \\ & w_1, w_2 \geq 0. \end{aligned}$$

The feasible regions of the primal and dual problems are given in Fig 11.2 and Fig 11.3 respectively.

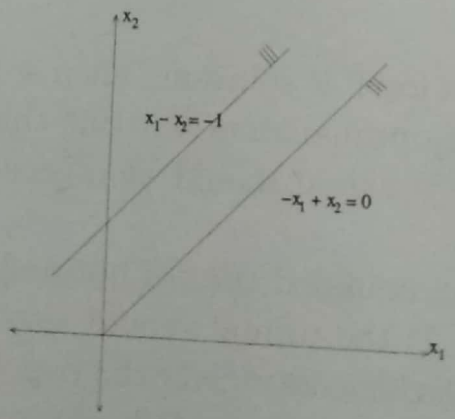


Fig. 4.2.

As we see here both feasible regions are empty, i.e. the primal and dual, both problems are infeasible.

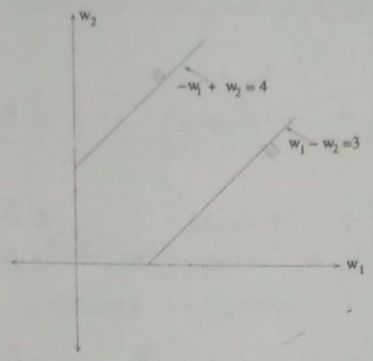


Fig. 4.3.

Theorem 4.4.4 (Complementary Slackness Theorem). *Let $\overline{x}$ and $\overline{w}$ be feasible solutions to the primal-dual pair (4.13)-(4.14). Then $\overline{x}$ and $\overline{w}$ are optimal to the respective problems (4.13) and (4.14) if and only if*

$$\overline{w}^T(A\overline{x} - b) = 0$$

and

$$\overline{x}^T(c - A^T\overline{w}) = 0.$$

The above conditions are called the complementary slackness conditions or in short the c.s.conditions.

Proof. We shall prove only the necessary part; the proof of sufficient part is similar. Thus assuming that $\overline{x}$ and $\overline{w}$ are optimal to the primal and the dual respectively, we have to show that the c.s.conditions hold.

Let $\alpha = \overline{w}^T(A\overline{x} - b)$ and $\beta = \overline{x}^T(c - A^T\overline{w})$. Then by the feasibility of $\overline{x}$ to the primal and the feasibility of $\overline{w}$ to the dual, we have $\alpha \leq 0$ and $\beta \leq 0$. Also

$$\begin{aligned}\alpha + \beta &= \overline{w}^T(A\overline{x} - b) + \overline{x}^T(c - A^T\overline{w}) \\ &= \overline{w}^TA\overline{x} - \overline{w}^Tb + \overline{x}^Tc - \overline{x}^TA^T\overline{w} \\ &= \overline{w}^TA\overline{x} - b^T\overline{w} + c^T\overline{x} - (\overline{w}^TA\overline{x})^T. \qquad (4.23)\end{aligned}$$

But $\overline{w}^TA\overline{x}$ is a scalar and therefore $\overline{w}^TA\overline{x} = (\overline{w}^TA\overline{x})^T$. Further, as $\overline{x}$ and $\overline{w}$ are optimal to the primal and dual respectively, by the strong duality theorem we have $c^T\overline{x} = b^T\overline{w}$. Hence from (4.23) we get that $\alpha + \beta = 0$. But then $\alpha \leq 0$, $\beta \leq 0$, $\alpha + \beta = 0$ imply that $\alpha = 0$ and $\beta = 0$, which prove that the c.s.conditions hold. □

4.5 A Convenient Way for Reading the Solution of the Dual

Consider the following primal LPP

$$\begin{aligned} &\text{Max} && c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ &\text{subject to} \\ & && a_{11}x_1 + \ldots + a_{1n}x_n && \le b_1 \\ & && \vdots \\ & && a_{r1}x_1 + \ldots + a_{rn}x_n && \le b_r \\ & && a_{r+1,1}x_1 + \ldots + a_{r+1,n}x_n && \ge b_{r+1} \\ & && \vdots \\ & && a_{r+s,1}x_1 + \ldots + a_{r+s,n}x_n && \ge b_{r+s} \\ & && a_{r+s+1,1}x_1 + \ldots + a_{r+s+1,n}x_n && = b_{r+s+1} \\ & && \vdots \\ & && a_{r+s+k,1}x_1 + \ldots + a_{r+s+k,n}x_n && = b_{r+s+k} \\ & && x_1 \ge 0, \ldots, x_n \ge 0. \end{aligned} \tag{4.24}$$

Here $r + s + k = m$ and, without any loss of generality, we can assume that the first r constraints are with '$\le$' sign, the next s constraints are with '$\ge$' sign and the next k constraints are with '$=$' sign. Also $b \ge 0$. The dual of problem (4.24) is

$$\begin{aligned} &\text{Min} && b^Tw \\ &\text{subject to} \\ & && A^Tw \ge c \\ & && w_1, w_2, \ldots, w_r \ge 0 \\ & && w_{r+1}, w_{r+2}, \ldots, w_{r+s} \le 0 \\ & && w_{r+s+1}, w_{r+s+2}, \ldots, w_{r+s+k} \text{ unrestricted in sign,} \end{aligned} \tag{4.25}$$

where $A = (a_{ij})$ is the matrix of order $(m \times n)$.

From the last section we know that if $\bar{x}$ is optimal to the primal (4.24) then $\bar{w}^T = c_B^T B^{-1}$ is optimal to (4.25) where c_B and B^{-1} are read from the last (optimal) simplex tableau of the primal (4.24). Here we may see that as the constraints in (4.24) are mixed, different components of w in (4.25) will be constrained to have different signs.

The main point which we wish to bring out here is that to find $\bar{w}$ we do not need to evaluate $c_B^T B^{-1}$ but rather read certain specific elements in the last row of the optimal simplex tableau of the primal.

For this, let us assume that problem (4.24) has been solved by using the Big-M method (we may make appropriate changes in the arguments if (4.24) is solved by the two phase method). Let us try to compute the number of columns which the simplex tableau will have. Obviously there will be n columns for the variables $x_1, x_2, \ldots, x_n$; r columns for the r slack variables x_{n+1}, $x_{n+2}, \ldots, x_{n+r}$; s columns for

the s surplus variables $x_{n+r+1}, x_{n+r+2}, \ldots, x_{n+r+s}$; s columns for the s artificial variables $x_{n+r+s+1}, x_{n+r+s+2}, \ldots, x_{n+r+2s}$ which have been added to these constraints; and k columns for the k artificial variables $x_{n+r+2s+1}, x_{n+r+2s+2}, \ldots, x_{n+r+2s+k}$ which have been added to k 'equal to' constraints. Therefore the (initial) simplex tableau for the problem (4.24) will look like

x_B	$\leftarrow n \rightarrow$	$\leftarrow r \rightarrow$	$\leftarrow s \rightarrow$	$\leftarrow s \rightarrow$	$\leftarrow k \rightarrow$
x_{B_1} x_{B_2} $\vdots$ x_{B_m}	*columns of A*	*slack columns*	*surplus columns*	*artificial columns for '$\geq$' type constraints*	*artificial columns for '$=$' type constraints*
$z(x_B)$	$\longleftarrow$	$\longleftarrow$	$z_j - c_j$	$\longrightarrow$	$\longrightarrow$

and at the optimality, i.e. in the last tableau, $(z_j - c_j) \geq 0, \quad j = 1, 2, \ldots, n,$
$n+1, \ldots, n+r, n+r+1, \ldots, n+r+s, n+r+s+1, \ldots, n+r+2s, n+r+2s+1, \ldots, n+r+2s+k.$

Now let us compute $\overline{w}_1$, i.e. the first component of the optimal dual vector $\overline{w}$. As $\overline{w}^T = c_B^T B^{-1}$, we have

$$\begin{aligned}
\overline{w}_1 &= \text{ First component of } c_B^T B^{-1} \\
&= c_B^T B^{-1} e_1, \; e_1 = \text{col } (1, 0, 0, \ldots, 0) \\
&= c_B^T (B^{-1} e_1) \\
&= z_{n+1} \\
&= (z_{n+1} - c_{n+1}) \; (\geq 0).
\end{aligned}$$

This is because $B^{-1}e_1$ is the y-column for the first slack variable x_{n+1}, i.e. $y^{(n+1)}$, $c_{n+1} = 0$ and $(z_j - c_j) \geq 0$ for all j; so in particular for $j = n+1$.

Similarly $\overline{w}_2 = (z_{n+2} - c_{n+2}) \geq 0, \ldots, \overline{w}_r = (z_{n+r} - c_{n+r}) \geq 0$. So the first r components of the optimal dual vector $\overline{w}$ are non-negative as desired.

Next let us compute $\overline{w}_{r+1}$, i.e. $(r+1)^{th}$ component of the vector $\overline{w}$. We have

$$\begin{aligned}
\overline{w}_{r+1} &= (r+1)^{th} \text{ component of } c_B^T B^{-1} \\
&= c_B^T B^{-1} e_{r+1} = -c_B^T B^{-1}(-e_{r+1}) \\
&= -c_B^T (B^{-1}(-e_{r+1})) \\
&= -z_{n+r+1} \\
&= -(z_{n+r+1} - c_{n+r+1}) \; (\leq 0).
\end{aligned}$$

This because $B^{-1}(-e_{r+1})$ is the y-column for the first surplus variable, i.e. $y^{(n+r+1)}$, $c_{n+r+1} = 0$ and $-(z_{n+r+1} - c_{n+r+1}) \leq 0$ because $z_{n+r+1} - c_{n+r+1} \geq 0$.

Similarly $\overline{w}_{r+2} = -(z_{n+r+2} - c_{n+r+2}) \leq 0, \ldots, \overline{w}_{r+s} = -(z_{n+r+s} - c_{n+r+s}) \leq 0$. So the next s components of the optimal dual vector $\overline{w}$ are non-positive as desired.

Next let us compute $\overline{w}_{r+s+1}$. We have

$$\begin{aligned}\overline{w}_{r+s+1} &= c_B^T B^{-1} e_{r+s+1} \\ &= c_B^T (B^{-1} e_{r+s+1}) \\ &= z_{n+r+2s+1}\end{aligned}$$

This is because $B^{-1}e_{r+s+1}$ is the y-column for the first artificial variable which has been added to '=' type constraints, i.e. $y^{(n+r+2s+1)}$ and $z_{n+r+2s+1}$ can have any sign.

Similarly $\overline{w}_{r+s+2} = z_{n+r+2s+2}, \ldots, \overline{w}_{r+s+k} = z_{n+r+2s+k}$ which are unrestricted as desired.

The above discussion can be summarized as follows

In the optimal dual solution $\overline{w}$,

(i) those components of w which are constrained to be '≥ 0', are obtained as the values of $(z_j - c_j)$ for r slack columns in the optimal simplex tableau.

(ii) those components of w which are constrained to be '≤ 0', are obtained as the values of $-(z_j - c_j)$ for s surplus columns in the optimal simplex tableau.

(iii) those components of w which are unrestricted in sign, are obtained as the values of z_j for k artificial columns (for the '=' constraints) in the optimal simplex tableau.

Therefore if the optimal simplex tableau of the primal is given then the optimal solution $\overline{w}$ of the dual can just be read from the last row as specified above rather than evaluating $c_B^T B^{-1}$.

Example 4.5.1 *Write the dual of the following (primal) LPP and solve the dual by solving the primal*

$$\begin{aligned} \textit{Max} \quad & -2x_1 - x_2 \\ \textit{subject to} \quad & \\ & 3x_1 + x_2 = 3 \\ & 4x_1 + 3x_2 \geq 6 \\ & x_1 + 2x_2 \leq 3 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Solution Using the rule table, we get the dual as

$$\begin{aligned} \text{Min} \quad & 3w_1 + 6w_2 + 3w_3 \\ \text{subject to} \quad & \\ & 3w_1 + 4w_2 + w_3 \geq -2 \\ & w_1 + 3w_2 + 2w_3 \geq -1 \\ & w_1 \in \mathbf{R}, w_2 \leq 0, w_3 \geq 0. \end{aligned}$$

Now we wish to obtain an optimal solution of the above problem (dual) from the optimal solution of the given LPP (primal). For this let us solve the given LPP by the Big-M method, and get the initial and final simplex tableaus as

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(a_1)}$	$y^{(a_2)}$	$y^{(4)}$
$x_{a_1} = 3$	3	1	0	1	0	0
$x_{a_2} = 6$	4	3	−1	0	1	0
$x_4 = 3$	1	2	0	0	0	1
$-9M$	$(2-7M)$	$(1-4M)$	M	0	0	0

and

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(a_1)}$	$y^{(a_2)}$	$y^{(4)}$
$x_1 = 3/5$	1	0	1/5	3/5	−1/5	0
$x_2 = 6/5$	0	1	−3/5	−4/5	3/5	0
$x_4 = 0$	1	0	1	1	−1	1
$-12/5$	0	1/5	1/5	$(M-2/5)$	$(M-1/5)$	0

respectively.

Therefore an optimal solution of the primal is $\bar{x}_1 = 3/5$, $\bar{x}_2 = 6/5$ and the optimal value is $-12/5$.

Now we wish to get an optimal solution of the dual by reading certain specific elements in the last row of the above tableau. For this we note that w_1 is unrestricted in sign and hence the value of $\overline{w}_1$ in the optimal dual solution $\overline{w}$ will be the value of z_j for that artificial variable which corresponds to '=' constraint.

Thus

$$\begin{aligned}\overline{w}_1 &= \text{ value of } z_j \text{ for the } y^{(a_1)} \text{ column}\\ &= (z_j - c_j) + c_j \text{ for the } y^{(a_1)} \text{ column}\\ &= (M - 2/5) + (-M)\\ &= -2/5\,.\end{aligned}$$

Also

$$\begin{aligned}\overline{w}_2 &= \text{ value of } -(z_j - c_j) \text{ for the surplus column } y^{(3)}\\ &= -1/5\,.\end{aligned}$$

$$\begin{aligned}\overline{w}_3 &= \text{ value of } (z_j - c_j) \text{ for the slack column } y^{(4)}\\ &= 0\,,\end{aligned}$$

and the optimal value of the dual is same as the optimal value of the primal, i.e. $-12/5$.

Remark 4.5.1 *Let us recollect the revised simplex method and note that the solution of the dual problem, namely* $\overline{w}^T = c_B^T B^{-1}$ *is automatically available in the first row of* B^{-1} *as stored in the revised simplex tableau.*

4.6 Some Useful Deductions from the Complementary Slackness Theorem

The complementary slackness conditions as stated in Theorem 4.4.4 provide certain useful information about the optimal solution of the dual (primal), knowing the optimal solution of the primal (dual). For this let us consider the first complementary slackness condition, namely $\overline{w}^T(A\overline{x} - b) = 0$, which can be expanded as

$$\sum_{i=1}^{m} \overline{w}_i \left(\sum_{j=1}^{n} a_{ij}\, \overline{x}_j - b_i \right) = 0. \tag{4.26}$$

But in (4.26), the finite sum of m non-positive quantities equals zero and hence we get

$$\overline{w}_i \left(\sum_{j=1}^{n} a_{ij}\, \overline{x}_j - b_i \right) = 0 \quad (i = 1, 2, \ldots, m). \tag{4.27}$$

Now for each i, the L.H.S. of (4.27) is the product of two numbers and therefore

$$\sum_{j=1}^{n} a_{ij}\, \overline{x}_j < b_i \Rightarrow \overline{w}_i = 0, \tag{4.28}$$

and

$$\overline{w}_i > 0 \Rightarrow \sum_{j=1}^{n} a_{ij}\, \overline{x}_j = b_i. \tag{4.29}$$

In view of the above relations we infer that if for the optimal solution $\overline{x}$ of the primal, the ith constraint holds as strict inequality, then in the optimal solution $\overline{w}$ of the dual, the ith component namely $\overline{w}_i$ must be zero. Equivalently if in the optimal solution $\overline{w}$ of the dual, $\overline{w}_i > 0$, then at the optimal point of the primal the ith primal constraint holds as an equation. Therefore (4.28)-(4.29) are also called complementary slackness conditions.

Economic Interpretation of the Complementary Slackness Conditions

If the primal problem (4.13) has certain meaningful economic interpretation then it is expected that the dual problem (4.14) and related duality results will also have some meaningful economic interpretations. In fact this is true and for details we may refer to an appropriate book on mathematical economics e.g. [63, 137, 80]. However, as an illustration we do present here an economic meaning of the complementary slackness conditions (4.28) and (4.29).

In the primal problem (4.13), let us interpret the parameters b, c and A as follows

b_i = units of the ith raw material available $(i = 1,2,\ldots,m)$
c_j = per unit profit for the jth product $(j = 1,2,\ldots,n)$
a_{ij} = units of ith raw material used in producing one unit of the jth product
$(i = 1,2,\ldots,m,\ j = 1,2,\ldots,n)$.

Thus we are given m different raw materials which can be used to produce n different products. These raw materials are available in limited supply and the entire output can be sold in the market. Our aim is to obtain the units of each product to be produced so that total profit is maximum. It is reasonably simple to see that the mathematical model of the above problem leads to problem (4.13).

Let us now look at the dual problem (4.14) in the context of the economic interpretation of problem (4.13) as described above. We note that the dimension of c_j is rupees per unit of the jth product and that of a_{ij} is units of the ith raw material per unit of the jth product. Therefore the constraints $\sum_{i=1}^{m} a_{ij}\, w_i \geq c_j$ imply that the dimension of w_i must be per unit of the ith raw material because in the given inequality the dimensions of both sides must be same.

Thus to each raw material i, there corresponds a dual variable w_i which gives the valuation (or price) of one unit of the ith raw material. In economics w_i are called the *shadow prices* and are different from the actual prevailing prices as explained below.

Let us consider the situation where it is decided that the raw materials on hand should be insured against fire, theft etc. This insurance is intended to protect the total income of the products after they are sold in the market. The problem is to find that insurance scheme which is large enough to provide the full compensation and at the same time minimizes the total insurance cost. It can be seen that the mathematical model of this later problem is precisely the dual problem (4.14).

The dual variables w_i are called *imputed values* or *shadow prices* for the various raw materials. As mentioned earlier these w_i are not the actual per unit costs of raw materials but rather what we really perceive about them. If we now look at the first complementary slackness condition

$$\sum_{j=1}^{n} a_{ij}\overline{x_j} < b_i \Rightarrow \overline{w}_i = 0\ ,$$

then we observe that if in our profit maximization problem, at the optimal production level, the ith raw material is not being used fully (i.e. $< b_i$) then that raw material should be a cheap (the way we perceive) raw material because in the optimal solution of the dual, $\overline{w}_i = 0$. On the other hand if the ith raw material is costly $(\overline{w}_i > 0)$ then in the optimal solution of the primal, the ith raw material will be consumed fully, i.e. $\sum_{j=1}^{n} a_{ij}\,\overline{x_j} = b_i$.

Alternate Algorithms for Solving LPP's

The complementary slackness theorem also opens up possibilities of developing new algorithms for solving LPP's. This is because it says that if we achieve the primal feasibility, the dual feasibility and the complementary slackness conditions then we have optimality for both primal and dual. As there are three things to be achieved at the end, we may think of algorithms which maintain any two of these throughout and stops when the third is also achieved.

In the simplex method we start from a primal feasible solution and make sure that it remains feasible for all iterations. We also maintain complementary slackness conditions throughout (why?) and stop when the dual feasibility is attained (i.e. all $(z_j - c_j) \geq 0$). Instead of this, we can think of an algorithm which starts from a dual feasible solution (i.e. all $(z_j - c_j) \geq 0$) and keeps it dual feasible throughout (i.e. all $(z_j - c_j) \geq 0$ for all iterations); maintains complementary slackness conditions throughout and stops when the primal feasibility is achieved (i.e. all $x_{B_i} \geq 0$). This, in view of the complementary slackness theorem, is certainly a valid algorithm for solving LPP's. This algorithm is called the *dual simplex method* which we shall study in Section 4.7. Another possibility could be to start from a basic solution which is neither primal feasible nor dual feasible and combine the simplex method and the dual simplex method suitably so that in the end all conditions of complementary slackness theorem are met. Such methods are called the *criss-cross methods*, which have been studied mainly by S. Zionts [172].

4.7 The Dual Simplex Method

We have already presented an intuitive idea of the dual simplex method in the last section. Here we present a stepwise description of the dual simplex algorithm which is due to C. E. Lemeke. We do not plan to give full mathematical justification of various results because they can be derived on the lines similar to the usual simplex method. The main point to note is that the dual simplex method is essentially equivalent to solving the dual of the given problem by the usual simplex method.

Now let the given LPP be

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax = b \\ & && x \geq 0, \end{aligned} \tag{4.30}$$

where $c \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $A = (a_{ij})_{m \times n}$ with $\text{rank} A = m$.

In (4.30), we do not insist that $b \geq 0$, because the starting point of the dual simplex algorithm is a basic solution $x_B = B^{-1}b$ for which all $(z_j - c_j) \geq 0$ but some x_{B_i} may be negative, i.e. the solution x_B is dual feasible but it may not be primal feasible. We now have the following steps

Step 1 Start with a primal basic solution $x_B = B^{-1}b$ (for some basis matrix B) for which all $(z_j - c_j) \geq 0$.

Step 2 Check if all $x_{B_i} \geq 0$. If the answer is 'yes' then we stop as x_B is optimal to the given LPP. If some $x_{B_i} < 0$ then we go to Step 3.

Step 3 Find the negative most value of x_{B_i}, i.e. $x_{B_r} = \underset{i}{Min}\{x_{B_i} : x_{B_i} < 0\}$. Then x_{B_r} becomes nonbasic and column $b^{(r)}$ of the variable x_{B_r} leaves the basis. The row corresponding to the variable x_{B_r} becomes the pivot row.

Step 4 Find

$$\frac{(z_k - c_k)}{y_{r_k}} = \underset{j}{Max}\left\{\frac{(z_j - c_j)}{y_{r_j}} : y_{r_j} < 0\right\}$$

then the variable x_k becomes a basic variable and the corresponding column $a^{(k)}$ enters the basis. The column corresponding to $(z_k - c_k)$ becomes the pivot column.

Step 5 Get the new dual simplex tableau by using the pivoting which is exactly same as in the usual simplex method, and then go to Step 2.

Here we make the following observations

(i) In the dual simplex method we first find the pivot row and then find the pivot column, whereas in the simplex method we first find the pivot column and then find the pivot row.

(ii) The pivot element in the dual simplex method is always negative, whereas in the simplex method it is always positive.

(iii) The maximum ratio criteria in the dual simplex method is used to make sure that all $(z_j - c_j)$ remain non-negative in the next iteration, whereas the minimum ratio criteria in the simplex method is used to guarantee that all x_{B_i} remain non-negative in the next iteration.

(iv) It is possible that in the pivot row there may not be any $y_{rj} < 0$. In that case it can be shown that the given LPP is infeasible. This situation is very similar to the usual simplex method where we know that if in the pivot column there is no $y_{ij} > 0$, then the given LPP has unbounded solution. As 'unboundedness' and 'infeasibility' are dual concepts, it is natural that the dual simplex method should check infeasibility because we know that the simplex method checks unboundedness of the given LPP.

(v) Steps 2 and 3 of the dual simplex method seem to be very natural but Step 4 probably needs some justification. We are given that for the current dual simplex tableau all $(z_j - c_j) \geq 0$. Once the pivot row has been identified the value of r gets fixed.

Then the basic aim of Step 4 is to guarantee that all $(\widehat{z_j} - c_j)$ remain non-negative for the next tableau as well. But

$$(\widehat{z_j} - c_j) = (z_j - c_j) - \frac{y_{rj}}{y_{rk}}(z_k - c_k)$$

and therefore $(\widehat{z_j} - c_j) \geq 0$ means that for all j,

$$(z_j - c_j) - \frac{y_{rj}}{y_{rk}}(z_k - c_k) \geq 0. \tag{4.31}$$

Now in (4.31), $(z_j - c_j) \geq 0$, $(z_k - c_k) \geq 0$, $y_{rk} < 0$ and therefore (4.31) trivially holds for all j for which $y_{rj} \geq 0$. Therefore we have to bother about those j for which $y_{rj} < 0$. In that case (4.31) gives

$$\frac{z_j - c_j}{y_{rj}} \leq \frac{z_k - c_k}{y_{rk}},$$

i.e.

$$\frac{z_k - c_k}{y_{rk}} = \underset{j}{Max}\left\{\frac{z_j - c_j}{y_{rj}} : y_{rj} < 0\right\},$$

which is nothing but the maximum ratio criteria as given in Step 4.

Example 4.7.1 *Use the dual simplex method to solve the following LPP*

Max $-2x_1 - x_2$

subject to

$$2x_1 - x_2 - x_3 \geq 3$$
$$x_1 - x_2 + x_3 \geq 2$$
$$x_1, x_2, x_3 \geq 0.$$

Solution We introduce the surplus variables x_4 and x_5 to get

Max $-2x_1 - x_2 + 0x_3 + 0x_4 + 0x_5$

subject to

$$2x_1 - x_2 - x_3 - x_4 = 3$$
$$x_1 - x_2 + x_3 - x_5 = 2$$
$$x_1, x_2, x_3, x_4, x_5 \geq 0.$$

Now taking the variables x_4 and x_5 as basic variables we obtain

$$x_B = \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} -3 \\ -2 \end{pmatrix}$$

$$y^{(1)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

$$y^{(2)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$y^{(3)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$y^{(4)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$y^{(5)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$c_B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad z(x_B) = 0$$

$$(z_1 - c_1) = 0 - (-2) = 2$$
$$(z_2 - c_2) = 0 - (-1) = 1$$
$$(z_3 - c_3) = 0 - 0 = 0$$
$$(z_4 - c_4) = 0 - 0 = 0$$
$$(z_5 - c_5) = 0 - 0 = 0.$$

As for the basic solution x_B, all $(z_j - c_j) \geq 0$, we have got the right situation to use the dual simplex method. The initial dual simplex tableau is

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
←	$x_4 = -3$	$\boxed{-2}$	1	1	1	0
	$x_5 = -2$	-1	1	-1	0	1
	0	2	1	0	0	0
		↑				

First iteration

Step 2 As not all $x_{B_i} \geq 0$, the current solution is not optimal.

Step 3 $x_{B_r} = \min\{x_{B_i} : x_{B_i} < 0\} = \min(-3, -2)$, which corresponds to x_4 and therefore x_4 becomes nonbasic variable.

Step 4 As in the pivot row only one $y_{rj} < 0$, we get the first column as the pivot column and x_1 becomes a basic variable.

Step 5 The pivoting element is $y_{kr} = -2$ and after pivoting we get the next tableau as

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
	$x_1 = 3/2$	1	$-1/2$	$-1/2$	$-1/2$	0
←	$x_5 = -1/2$	0	1/2	$\boxed{-3/2}$	$-1/2$	1
	-3	0	2	1	1	0
				↑		

Second iteration

The current solution is still not optimal as $x_5 = -1/2 < 0$. So we identify the pivot row and conclude that x_5 becomes a nonbasic variable. To identify the pivot column we use the maximum ratio criteria, i.e.

$$Max\left(\frac{1}{-3/2}\ \frac{1}{-1/2}\right) = -2/3,$$

which corresponds to the value $(z_3 - c_3)$ and therefore x_3 becomes a basic variable. We also note that the pivot element is $-3/2$. After pivoting, we get the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
$x_1 = 5/3$	1	$-2/3$	0	$-2/3$	$-1/3$
$x_3 = 1/3$	0	$-1/3$	1	$1/3$	$-2/3$
$-10/3$	0	$5/3$	0	$2/3$	$2/3$

As all $x_{B_i} \geq 0$, we obtain the optimal solution as $(x_1^* = 5/3, x_2^* = 0, x_3^* = 1/3)$ and the optimal value as $(-10/3)$.

Example 4.7.2 *Use the dual simplex method to check that the following LPP is infeasible*

$$\begin{aligned} &Max && -x_1 \\ &subject\ to \end{aligned}$$

$$\begin{aligned} x_1 - x_2 &\geq 3 \\ -x_1 + x_2 &\geq 4 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Solution Taking x_3 and x_4 as the surplus variables we get the following dual simplex tableaus

	x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
	$x_3 = -3$	-1	1	1	0
$\leftarrow$	$x_4 = -4$	1	$\boxed{-1}$	0	1
	0	1	0	0	0
			$\uparrow$		

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$
$x_3 = -7$	0	0	1	1
$x_2 = 4$	-1	1	0	-1
0	1	1	0	0

Now $x_3 < 0$ but in that row, no $y_{rj} < 0$, hence the given LPP is infeasible. Fig 11.3 verifies the same thing graphically.

Remark 4.7.1 *In Example 4.7.1, if we take the objective function as* $(2x_1 - x_2)$, *then* $(z_1 - c_1) = -2$ *and therefore the dual simplex method is no more applicable. In these situations there does not seem to be a way to find a starting solution of the dual simplex method. Though there are some methods in the literature to find a starting solution for a general LPP which is required to be solved by the dual simplex method, they are not very attractive because the artificial variable technique of the usual simplex method is generally more simple. It is only when the given LPP is in the minimization form with* $c \geq 0$, *all constraints are with '*$\geq$*' sign and* $b \geq 0$, *the dual simplex method has major advantage over the simplex method. However the dual simplex method is extremely useful in another way as it becomes a very handy tool for studying post optimality analysis and integer linear programming. While post optimality analysis is a topic of discussion for the next section, integer linear programming is studied later in Chapter 6.*

4.8 Post Optimality Analysis

The optimal solution and the optimal value of a given LPP generally depend upon the problem parameters, namely c, b and A. Let $\text{LP}(c, b, A)$ denotes the linear programming problem for a given c, b and A.

In post optimality analysis we are given that the problem $\text{LP}(c, b, A)$ has been solved by the simplex method, i.e. $x^\star = (x_B = B^{-1}b, x_R = 0)$ and $z(x_B) = c_B^T x_B$ are known together with the optimal simplex tableau for the basic matrix B. Later we are told that at the modeling stage, these parameters were not correctly specified; they should have been taken as $\hat{c}$, $\hat{b}$ and $\hat{A}$ respectively. In other words we were supposed to solve the problem $\text{LP}(\hat{c}, \hat{b}, \hat{A})$ whereas we have actually solved the problem $\text{LP}(c, b, A)$. What is the best way out now? Obviously we do not want to solve the problem $\text{LP}(\hat{c}, \hat{b}, \hat{A})$ from the very beginning and therefore wish to see if we can do something better. Can we really start from the optimal simplex tableau of $\text{LP}(c, b, A)$ itself and make appropriate modifications so as to get an optimal solution of $\text{LP}(\hat{c}, \hat{b}, \hat{A})$? The post optimality analysis aims to provide answers to these questions in a limited way and the dual simplex method becomes the basic tool to be employed.

In our presentation here we limit ourselves to the following type of changes in the problem parameters

(i) change in the resource vector b
(ii) change in the cost vector c
(iii) addition of a new constraint
(iv) addition of a new variable
(v) deletion of a constraint
(vi) deletion of a variable
(vii) change in the coefficient matrix A.

We shall now discuss each of the above cases and explain the working through the below given example only.

Consider the linear programming problem

$$\begin{aligned} &\text{Max} && z = 3x_1 + 4x_2 + x_3 + 7x_4 \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} 8x_1 + 3x_2 + 4x_3 + x_4 &\le 7 \\ 2x_1 + 6x_2 + x_3 + 5x_4 &\le 3 \\ x_1 + 4x_2 + 5x_3 + 2x_4 &\le 8 \\ x_1, x_2, x_3, x_4 &\ge 0. \end{aligned} \tag{4.32}$$

Let the above problem be solved by the simplex method. The optimal simplex tableau is

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$x_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0
$x_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0
$x_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1
83/19	0	169/38	1/2	0	29/38	53/38	0

Thus the optimal solution of the given LPP is $(x_1^* = 16/19, x_2^* = 0, x_3^* = 0, x_4^* = 5/19)$ and the optimal value is $z^* = 4.3$.

(i) **Change in the Resource Vector** b

We shall first discuss the change in the resource vector b. Let the given vector b be changed to a new vector, say, $\hat{b}$. The effect of b can be seen in the computation of the b.f.s x_B, which is given by, $x_B = B^{-1}b$. Here, it may be noted that only the current solution x_B will change with the change in b. Therefore if b is changed to $\hat{b}$, then the current b.f.s x_B will change to $\hat{x}_B$ where $\hat{x}_B = B^{-1}\hat{b}$. If $\hat{x}_B \ge 0$, then the original basis continues to be feasible for the new problem and $\hat{x}_B$ becomes the new optimal solution with the new value of the objective function as $z(\hat{x}_B)$. But, if some component of $\hat{x}_B$ is less than zero, then the original basis is no longer feasible for the new problem and hence we have to use the dual simplex method to find the new optimal basic feasible solution.

Let us examine the change in the optimal solution if $b = (7,3,8)^T$ is changed to $\hat{b} = (13,3,8)^T$ in the above example.

For this, we first identify the current basis inverse, i.e. B^{-1}. Let us recall that the inverse of the current basis matrix is always available in the optimal simplex tableau precisely at those locations where we had identity columns in the initial simplex tableau. Therefore, for our example

$$B^{-1} = \begin{pmatrix} 5/38 & -1/38 & 0 \\ -1/19 & 4/19 & 0 \\ -1/38 & -15/38 & 1 \end{pmatrix},$$

because these are the columns corresponding to the slack variables in the optimal simplex tableau. Now,

$$\hat{x}_B = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_4 \\ \hat{x}_7 \end{bmatrix} = B^{-1}\hat{b} = \begin{pmatrix} 5/38 & -1/38 & 0 \\ -1/19 & 4/19 & 0 \\ -1/38 & -15/38 & 1 \end{pmatrix} \begin{pmatrix} 13 \\ 3 \\ 8 \end{pmatrix} = \begin{pmatrix} 31/19 \\ -1/19 \\ 123/19 \end{pmatrix}.$$

Here, we observe that $\hat{x}_4$ takes the negative value and hence the new basic solution $\hat{x}_B$ is not feasible. Therefore, we use the dual simplex method to obtain a new optimal b.f.s and get the following tableaus

$\hat{x}_B$	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$\hat{x}_1 = 31/19$	1	9/38	1/2	0	5/38	−1/38	0
$\hat{x}_4 = -1/19$	0	21/19	0	1	−1/19	4/19	0
$\hat{x}_7 = 123/19$	0	59/38	9/2	0	−1/38	−15/38	1
87/19	0	169/38	1/2	0	29/38	53/38	0

$\hat{x}_B$	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$\hat{x}_1 = 3/2$	1	3	1/2	5/2	0	1/2	0
$\hat{x}_5 = 1$	0	−21	0	−19	1	−4	0
$\hat{x}_7 = 13/2$	0	1	9/2	−1/2	0	−1/2	1
9/2	0	5	1/2	1/2	0	3/2	0

Thus the optimal solution of the new problem is $(x_1^* = 3/2, x_2^* = 0, x_3^* = 0, x_4^* = 0)$ and the optimal value is $z^* = 4.5$.

(ii) Change in the Cost Vector c

Let us next discuss the change in the cost vector c. In this situation there may arise two cases.

Case 1: The change is in c_j where j belongs to the set of indices of the non-basic vectors, i.e. the change in the cost vector corresponds to the non-basic variable only. Such a change will not affect the objective function value, $z(x_B) = c_B^T x_B$, as this does not involve c_j. Infact the only quantities that may change are $(z_j - c_j)$. Thus the change in c_j, denoted by $\hat{c}_j$, will affect $z_j - \hat{c}_j$ value, as $z_j - \hat{c}_j = c_B^T y_j - \hat{c}_j$. If this value is less than zero, then the simplex method is used to obtain the new optimal solution, otherwise, there is no change in the optimal value and the optimal solution.

Let us study the effect of change in c from $(3,4,1,7)^T$ to $\hat{c} = (3,8,1,7)^T$ in the above example.

Here, the change in c is with respect to the non-basic variable and hence the updated tableau is as follows

$\hat{x}_B$	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$\hat{x}_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0
$\hat{x}_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0
$\hat{x}_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1
83/19	0	17/38	1/2	0	29/38	53/38	0

Since all $z_j - c_j \geq 0$ for all j, therefore, this change in c has not affected the optimal solution and the optimal value.

Case 2: The change is in c_j where j belongs to the set of indices of the basic vectors, i.e. the change in the cost vector corresponds to the basic variables. Let $\hat{c}_B$ be the changed cost vector. In this case the objective function value $z(x_B) = c_B^T x_B$ and also the relative cost coefficients $(z_j - c_j) = c_B^T y^{(j)} - c_j$, all may change. Therefore, we compute the new objective value $z(\hat{x}_B) = \hat{c}_B^T \hat{x}_B$ and new relative cost coefficients $\hat{z}_j - c_j = \hat{c}_B^T y_j - c_j$ for all j belonging to the set of non-basic variables. Now, similar to Case 1, if for some j, $\hat{z}_j - c_j < 0$, then the simplex method will be used to obtain the new optimal solution, otherwise the current b.f.s will remain optimal.

Let us study the effect of change in c from $(3,4,1,7)^T$ to $(2,4,1,8)^T$ in the above example.

Here, the change in c is with respect to the basic variables and hence we compute new value of $z(\hat{x}_B)$ and $(\hat{z}_j - c_j)$ to get the updated tableau as follows

$\hat{x}_B$	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$\hat{x}_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0
$\hat{x}_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0
$\hat{x}_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1
83/19	0	202/38	0	0	−6/38	62/38	0

Note that in this case we need to compute $\hat{z}_j - c_j$ for all non-basic variables only because for basic variables it has to be zero. Now, in the updated tableau $\hat{z}_5 - c_5 < 0$ and hence we use the simplex method to obtain the new optimal solution.

(iii) **Addition of a New Constraint**

Here again there are two cases to be discussed because the current b.f.s may or may not satisfy the given additional constraint.

Case 1 If the new constraint is satisfied by the current optimal solution x^*, then the current solution remains optimal for the new problem as well. Let us discuss the effect of addition of a new constraint with respect to problem (4.32). Let the added constraint be $2x_1 + 3x_2 + x_3 + 5x_4 \leq 4$. As the current optimal solution $x_1^* = 16/19$, $x_2^* = 0$, $x_3^* = 0$, $x_4^* = 5/19$ satisfies the new constraint

$2x_1 + 3x_2 + x_3 + 5x_4 \leq 4$, because $2 \times 16/19 + 5 \times 5/19 = 57/19 = 3 < 4$, the optimal solution remains unchanged.

Case 2 If the additional constraint is violated by the current optimal solution x^*, then the problem has to be solved further by taking the new constraint into consideration. Each new constraint in the form of an inequality gives rise to a slack variable and if necessary artificial variable also. For the constraints in the form of equations, artificial variables are introduced. The problem is then solved by the dual simplex method. Here it must be noted that the new additional constraint can not be taken directly in the optimal simplex tableau of the given problem, but rather it has to be first expressed in the canonical form.
Let us study the effect on the optimal solution after introducing the additional constraint $2x_1 + 3x_2 + x_3 + 5x_4 \geq 4$.

Here, the current solution violates the new constraint. Therefore in order to obtain the new solution, we first express the given constraint in the canonical form and for that we write it as

$$2x_1 + 3x_2 + x_3 + 5x_4 - x_8 = 4\,, \tag{4.33}$$

where $x_8 \geq 0$ is the surplus variable.
Now the above equation has to be expressed in the canonical form, i.e. it should have only one basic variable with coefficient as 'one' and all other variables in the equation must be nonbasic variables. For our example, we need to make x_8 a basic variable and express the other basic variables, namely, x_1 and x_4, in terms of the nonbasic variables. But from the first two rows of the current optimal simplex tableau we have

$$x_1 = 16/19 - (9/38x_2 + 1/2x_3 + 5/38x_5 - 1/38x_6) \tag{4.34}$$

and

$$x_4 = 5/19 - 21/19x_2 + 1/19x_5 - 4/19x_6. \tag{4.35}$$

Substituting for x_1 and x_4 from (4.34) and (4.35) in (4.33) we get

$$-1 = x_8 + 3x_2 - x_6 \tag{4.36}$$

Introducing (4.36) in the given tableau we get

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$	$y^{(8)}$
$x_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0	0
$x_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0	0
$x_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1	0
$x_8 = -1$	0	3	0	0	0	−1	0	1
83/19	0	169/38	1/2	0	29/38	53/38	0	0

Now by employing the dual simplex method we can obtain the optimal solution of the new problem.

(iv) **Addition of a New Variable**

Since addition of the new variable is dual to the addition of a new constraint, this case can obviously be handled by the usual simplex method.
Let the new variable to be added be x_{n+1}. Also let $a_{i,n+1}$ $(i = 1, 2, \ldots, m)$ and c_{n+1} be its coefficients in the constraints and and the objective function, respectively. Since the number of constraints remain the same, the original optimal solution remains basic feasible for the new problem as well. Therefore we only calculate $z_{n+1} - c_{n+1}$. Obviously the evaluation of $z_{n+1} - c_{n+1}$ will need the evaluation of vector $y^{(n+1)}$. If $z_{n+1} - c_{n+1}$ is non negative, then the current optimal solution remains optimal for the new problem. If not, then we use the simplex method to obtain a new optimal solution.
We study the effect of introducing a new variable x_8 with the cost coefficient 4 and the activity vector $(3, -2, 4)^T$ with respect to problem (4.32). Since we need to evaluate $(z_8 - c_8)$, we first find

$$y^{(8)} = \begin{pmatrix} 5/38 & -1/38 & 0 \\ -1/19 & 4/19 & 0 \\ -1/38 & -15/38 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ -2 \\ 4 \end{pmatrix} = \begin{pmatrix} 17/38 \\ -11/19 \\ 179/38 \end{pmatrix}.$$

and this gives $z_8 - c_8 = c_B^T y^{(8)} - c_8 = -225/38$. Therefore the updated tableau is

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$	$y^{(8)}$
$x_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0	17/38
$x_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0	−11/19
$x_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1	179/38
83/19	0	169/38	1/2	0	29/38	53/38	0	−255/38

Now the simplex method can be used to find the new optimal solution.

(v) Deletion of a Variable

Here we need to consider two cases.

Case 1 If the deleted variable is a non-basic variable or with zero value in the optimal basis, then the original solution remains optimal because the zero value of the variable in the optimal solution makes the variable nonexistent in effect.

Case 2 If the variable to be deleted is present in the basis with positive value, then its removal will affect the solution. To obtain the new solution we first make this basic variable to leave from the basis forcefully, i.e. we replace this basic variable with some non-basic variable and make it a non-basic variable. Once it becomes non-basic variable, it can be removed from the tableau as it has zero value and its removal does not change the optimal value.

We consider problem (4.32) and study the effect of deletion of variable x_4, and hence find the new b.f.s if it exists. As x_4 is a basic variable, its removal will affect the optimal solution. Therefore as discussed, we first make it nonbasic and then delete it from the basis. Hence we replace the variable x_4 with x_2 to get the following tableau

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$x_1 = 11/14$	1	0	1/2	−3/14	1/7	−1/14	0
$x_2 = 5/21$	0	1	0	19/21	−1/21	4/21	0
$x_7 = 263/42$	0	0	9/2	−59/42	−1/21	−29/42	1
43/14	0	169/38	1/2	0	29/38	53/38	0

Hence the new b.f.s is $(x_1^* = 11/14, x_2^* = 5/21, x_3^* = 0, x_4^* = 0)$.

(vi) Deletion of a Constraint

If the constraint to be deleted is such that its slack variable has a positive value in the optimal solution then its deletion leaves the optimal solution unchanged. This is because the constraint is not being satisfied as an equality by the optimal solution and it is ineffective in determining the optimal solution. Therefore the ineffective constraint may be deleted from the problem without changing the optimal value. These constraints are known as inactive constraints.

However, if the constraint to be deleted is such that its slack variable has zero value in the optimal solution i.e. the constraint is an active constraint, then its deletion will change the optimal value. In this situation we first make the active constraint inactive by introducing corresponding slack variable in the basis and then delete the inactive constraint. We will explain the above procedure with the help of problem (4.32).

Let us study the effect of removal of the constraint $8x_1 + 3x_2 + 4x_3 + x_4 \leq 7$. From the optimal simplex tableau, we observe that the variable x_5 is the slack variable corresponding to the above constraint and it is not in the basis. Thus the given constraint is an active constraint. So we make it inactive and then drop it and for that we insert x_5 in the basis replacing the basic variable x_1. This gives the following updated tableaus

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$x_1 = 16/19$	1	9/38	1/2	0	5/38	−1/38	0
$x_4 = 5/19$	0	21/19	0	1	−1/19	4/19	0
$x_7 = 126/19$	0	59/38	9/2	0	−1/38	−15/38	1
83/19	0	169/38	1/2	0	29/38	53/38	0

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$
$x_5 = 32/5$	38/5	9/5	19/5	0	1	−1/5	0
$x_4 = 1/5$	2/5	6/5	1/5	1	0	1/5	0
$x_7 = 34/5$	1/5	8/5	23/5	0	0	−2/5	1
21/5	14/5	22/5	2/5	0	0	7/5	0

In the above tableau the slack variable corresponding to the first constraint has taken the positive value and hence this constraint has become inactive and therefore can be deleted from the tableau. To obtain the new optimal solution now we need to use the usual simplex method.

(vii) **Change in the coefficient matrix A**

Here again two cases can arise.

Case 1 If the vector $a^{(j)}$ is changed to $\hat{a}^{(j)}$ where x_j is a non-basic variable in the original optimal solution, then the modified $(\hat{z}_j - c_j)$ can be computed as $c_B^T \hat{a}_j - c_j$. If $\hat{z}_j - c_j \geq 0$ for all j, then the optimal solution remains unchanged. But if for some j, $\hat{z}_j - c_j < 0$, then further iterations of the simplex method may be performed to find the new optimal solution.

Case 2 If the changes are in $a^{(k)}$ where x_k is a basic variable of the original optimal solution, then we introduce a new variable $\hat{x}_k$ in the system with coefficient $\hat{a}^{(k)}$ and cost $\hat{c}_k = c_k$. In this new problem we treat x_k as an artificial variable, i.e. assign large negative cost with it and eliminate it from the basis, and then introduce $\hat{x}_k$ in the basis.

Let us study the effect of change of values of a_{21}, a_{22}, a_{23} from $(3, 6, 4)^T$ to $(3, 6, 1)^T$. The changes are in the coefficients of x_2 which is a nonbasic variable. Therefore the updated relative cost is given by

$$\hat{z}_2 - c_2 = c_B^T B^{-1} \hat{a}_2 - c_2$$

$$= \begin{pmatrix} 3 & 1 & 0 \end{pmatrix} \begin{pmatrix} 5/38 & -1/38 & 0 \\ -1/19 & 4/19 & 0 \\ -1/38 & -15/38 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 6 \\ 1 \end{pmatrix} - 4$$

$$= -2.184$$

Since the value of $\hat{z}_2 - c_2$ is less than zero, therefore the simplex method can be used to find the new optimal solution.

Next let us study the change in the coefficient matrix corresponding to a basic variable. Let us change the values of a_{11}, a_{12}, a_{13} from $(8,2,1)^T$ to $(8,3,1)^T$. Since the change is in the coefficients of x_1 which is a basic variable in the optimal solution, to study the effect we introduce a new variable say $\hat{x}_1$ with coefficients $(8,3,1)^T$ and the coefficients in the objective function as $\hat{c}_1 = 3$. Here we assign $-M$ cost to x_1 and perform Big-M method by treating x_1 as an artificial variable, and introducing $\hat{x}_1$ as a new variable.

x_B	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$	$y^{(8)}$
$x_1 = \frac{16}{19}$	1	$\frac{9}{38}$	$\frac{1}{2}$	0	$\frac{5}{38}$	$\frac{-1}{38}$	0	$\frac{37}{38}$
$x_4 = \frac{5}{19}$	0	$\frac{21}{19}$	0	1	$\frac{-1}{19}$	$\frac{4}{19}$	0	$\frac{4}{19}$
$x_7 = \frac{126}{19}$	0	$\frac{59}{38}$	$\frac{9}{2}$	0	$\frac{-1}{38}$	$\frac{-15}{38}$	1	$\frac{-15}{38}$
$\frac{5}{19} - M$	0	$\frac{-9M+294}{38}$	$\frac{-M-2}{2}$	0	$\frac{-5M-14}{38}$	$\frac{56+M}{38}$	0	$\frac{-37M+76}{38}$

Since the value of $\hat{z}_8 - c_8$ is negative most, therefore the simplex method can be used to find the new optimal solution.

4.9 Two Person Zero-Sum Matrix Games

In this section, we present certain basic definitions and preliminaries with regard to two person zero-sum matrix games.

Let $\mathbf{R}^n$ denote the n-dimensional Euclidean space and $\mathbf{R}^n_+$ be its non-negative orthant. Let $A \in R^{m \times n}$ be an $(m \times n)$ real matrix and $e^T = (1,1,\ldots,1)$ be a vector of 'ones' whose dimension is specified as per the specific context. By a two person zero-sum matrix game G we mean the triplet $G = (S^m, S^n, A)$ where $S^m = \{x \in \mathbf{R}^m_+, e^T x = 1\}$ and $S^n = \{y \in \mathbf{R}^n_+, e^T y = 1\}$. In the terminology of the matrix game theory, S^m (respectively S^n) is called the *strategy space* for *Player I* (respectively *Player II*) and A is called the *pay-off matrix*. Then, the elements of S^m (respectively S^n) which are of the form $x = (0,0,\ldots,1,\ldots,0)^T = e_i$, where 1 is at i^{th} place (respectively $y = (0,0,\ldots,1,\ldots,0)^T = e_j$, where 1 is at

the jth place) are called *pure strategies* for Player I (respectively Player II). If Player I chooses i^{th} pure strategy and Player II chooses j^{th} pure strategy then a_{ij} is the amount paid by Player II to Player I. If the game is zero-sum then $-a_{ij}$ is the amount paid by Player I to Player II i.e. the gain of one player is the loss of other player. The quantity $E(x, y) = x^T Ay$ is called the *expected pay-off* of Player I by Player II, as elements of S^m (respectively S^n) can be thought of as a set of all probability distributions over $I = \{1, 2, \ldots, m\}$ (respectively $J = \{1, 2, \ldots, n\}$). It is customary to assume that Player I is a maximizing player and Player II is a minimizing player. The triplet $PG = (I, J, A)$ is called the *pure form* of the game G whenever G is being referred as the *mixed extension* of the pure game G. We shall refer to a two person zero-sum game always as $G = (S^m, S^n, A)$ and if the game is in the pure form it will be clear from the context itself. Thus, for us S^m refers to the (mixed) strategy space of Player I, S^n refers to the (mixed) strategy space of Player II, and A refers to the pay-off matrix which introduces the function $E : S^m \times S^n \to \mathbf{R}$ given by $E(x, y) = x^T Ay$, called the *expected pay-off function.*

The meaning of the solution of the game $G = (S^m, S^n, A)$ is best understood in terms of maxmin and minmax principles for Player I and Player II respectively. According to this principle, each player adopts that strategy which results in the best of the worst outcomes. In other words, Player I (the maximizing player) decides to play that strategy which corresponds to the maximum of the minimum gain for his different courses of action. This is known as the *maxmin principle.*

Similarly, Player II (the minimizing player) also likes to play safe and in that case he selects that strategy which corresponds to the minimum of the maximum losses for his different courses of action. This is known as the *minmax principle.*

Employing the maxmin principle for Player I, we obtain $\underline{v} = \underset{x \in S^m}{MaxMin}\ \underset{y \in S^n}{} (x^T Ay)$, called the *lower value* of the game. Similarly the minmax principle for Player II gives $\overline{v} = \underset{y \in S^n}{MinMax}\ \underset{x \in S^m}{} (x^T Ay)$, called the *upper value* of the game. It is well known that $\overline{v} \geq \underline{v}$. The main result of two-person zero-sum matrix game theory asserts that, in fact, these are equal, i.e $\overline{v} = \underline{v} = v^*$, which is then called the *value of the game.* The following theorem is very useful in this regard.

Theorem 4.9.1 *If there exists* $(x^*, y^*, v^*) \in S^m \times S^n \times \mathbf{R}$ *such that*

(i) $E(x^*, y) \geq v^*$, $\forall\ y \in S^n$, *and*
(ii) $E(x, y^*) \leq v^*$, $\forall\ x \in S^m$,

then $\overline{v} = v^* = \underline{v}$ *and conversely.*

Definition 4.9.1 (Saddle point). *Let* $E : S^m \times S^n \longrightarrow \mathbf{R}$ *be given by* $E(x, y) = x^T Ay$. *The function* E *is said to have a saddle point* (x^*, y^*) *if* $E(x^*, y) \geq E(x^*, y^*) \geq E(x, y^*)$, $\forall\ x \in S^m$ *and* $\forall\ y \in S^n$.

In view of the above definition we have the following corollary for Theorem 4.9.1.

Corollary 4.9.1 *A necessary and sufficient condition that* $\overline{v} = \underline{v}$ *i.e.* $\underset{y\in S^n}{MinMax}\ \underset{x\in S^m}{}\ (x^TAy) = \underset{x\in S^m}{MaxMin}\ \underset{y\in S^n}{}\ (x^TAy)$, *is that the function* $E(x, y)$ *has a saddle point* (x^*, y^*). *Here* $v^* = E(x^*, y^*) = \underline{v} = \overline{v}$.

Theorem 4.9.1 leads to the following definition of the solution of the game G.

Definition 4.9.2 (Solution of a game). *Let* $G = (S^m, S^n, A)$ *be the given game. A triplet* $(x^*, y^*, v^*) \in S^m \times S^n \times \mathbf{R}$ *is called a solution of the game G if*

$$E(x^*, y) \geq v^*, \quad \forall\ y \in S^m,$$

and

$$E(x, y^*) \leq v^*, \quad \forall\ x \in S^n.$$

Here x^* *is called an optimal strategy for Player I,* y^* *is called an optimal strategy for Player II, and* v^* *is called the value of the game G.*

Remark 4.9.1. In view of Theorem 4.9.1 and its Corollary 4.9.1, (x^*, y^*, v^*) is a solution of the game G if and only if (x^*, y^*) is a saddle point of E and in that case $v^* = E(x^*, y^*)$. Such a saddle point is guaranteed to exist if $\underline{v} = \overline{v}$ and conversely. Here it may be noted that only the existence of $(\overline{x}, \overline{y}) \in S^m \times S^n$ such that $\underset{y\in S^n}{MinMax}\ \underset{x\in S^m}{}\ (x^TAy) = \underset{x\in S^m}{MaxMin}\ \underset{y\in S^n}{}\ (x^TAy) = \overline{x}^TA\overline{y}$, is not a sufficient condition in order that $(\overline{x}, \overline{y})$ be a solution of the matrix game G, i.e. this may not imply that $(\overline{x}, \overline{y})$ constitutes an optimal pair of strategies. For example, if $G = (S^2, S^2, A)$ with $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ then $\underline{v} = \overline{v} = 1$. Also $x^* = \left(\frac{1}{2}, \frac{1}{2}\right)^T = y^*$ constitutes a saddle point of E and therefore a pair of optimal strategies. However $\overline{x} = \left(\frac{1}{2}, \frac{1}{2}\right)^T$, $\overline{y} = (1, 0)^T$ also gives $E(\overline{x}, \overline{y}) = 1$, but $\overline{y}$ is obviously not optimal to Player II. The main reason being that (x^*, y^*) is a saddle point of $E(x, y)$ but $(\overline{x}, \overline{y})$ is not.

Next we answer the basic question regarding the existence of a solution for the game G. The following theorem is very fundamental in this context as it asserts that every two-person zero-sum matrix game G always has a solution.

Theorem 4.9.2 (Fundamental theorem of matrix games). *Let* $G = (S^m, S^n, A)$. *Then* $\underset{y\in S^n}{MinMax}\ \underset{x\in S^m}{}\ (x^TAy)$ *and* $\underset{x\in S^m}{MaxMin}\ \underset{y\in S^n}{}\ (x^TAy)$ *both exist and are equal.*

Here the problem $\underset{y\in S^n}{MinMax}\ \underset{x\in S^m}{}\ (x^TAy)$ (respectively $\underset{x\in S^m}{MaxMin}\ \underset{y\in S^n}{}\ (x^TAy)$ is called Player I's (respectively Player II's) problem. If there exists $(i_o, j_o) \in I \times J$ such that $a_{i_o,j} \geq a_{i_o,j_o}$ for all i and j then (i_o, j_o) is called a *pure saddle point* and in that case we say that the game G has a solution in the pure form. In this situation

$$\left(\underset{j\in J}{MinMax}\ \underset{i\in I}{}\ a_{ij}\right) = \left(\underset{i\in I}{MaxMin}\ \underset{j\in J}{}\ a_{ij}\right) = a_{i_o,j_o}$$

and therefore i_o gives an optimal pure strategy for Player I, j_o gives an optimal pure strategy for Player II, and a_{i_o,j_o} becomes the value of the game G. In this case it may be

noted that a_{i_o,j_o} is the smallest element in the i_o^{th} row and the largest element in the j_o^{th} column.

Thus Theorem 4.9.2 above guarantees that every two person zero-sum matrix game G has a solution. If there is no solution in the pure form then there is certainly a solution in the mixed form. Therefore, the question 'How to obtain the solution for this matrix game G?', is to be addressed in the next section.

4.10 Linear Programming and Matrix Game Equivalence

We shall now establish an equivalence between two person zero-sum matrix game $G = (S^m, S^n, A)$ and a pair of primal-dual linear programming problems. This equivalence besides being interesting mathematically, is also very useful as it provides a very efficient way to solve the given game G.

Let us consider the Player I's (respectively Player II's) problem: $\underset{y\in S^n}{Min}\ \underset{x\in S^m}{Max}\ (x^TAy)$ (respectively $\underset{x\in S^m}{Max}\ \underset{y\in S^n}{Min}\ (x^TAy)$). Since S^m and S^n are compact convex sets and for a given x (respectively given y), the function $E(x,y)$ is a linear function of y (respectively x), the $\underset{y\in S^n}{Min}\ x^TAy$ (respectively $\underset{x\in S^m}{Max}\ (x^TAy)$) will be attained at an extreme point of S^n (respectively S^m). Therefore for a given $x \in S^m$,

$$\underset{y\in S^n}{Min}\ (x^TAy) = \underset{1\le j\le n}{Min}\ (x^TAe_j),$$

where $e_j = (0,0,\ldots,1,\ldots,0)^T$ with '1' at the jth place, is the jth pure strategy of Player II. Thus

$$\underset{x\in S^m}{Max}\ \underset{y\in S^n}{Min}\ (x^TAy) = \underset{x\in S^m}{Max}\ \underset{1\le j\le n}{Min}\left(\sum_{i=1}^{m} a_{ij}x_i\right).$$

If we now take $v = \underset{1\le j\le n}{Min}\left(\sum_{i=1}^{m} a_{ij}x_j\right)$, then the maxmin value for Player I is obtained by solving the following linear programming problem

$$\text{Max} \quad v$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} a_{ij}x_i &\ge v \quad (j = 1,2,\ldots,n) \\ e^Tx &= 1 \\ x &\ge 0. \end{aligned} \tag{4.37}$$

Similarly the minmax value for Player II is obtained as a solution of the following linear programming problem

Min w
subject to

$$\sum_{j=1}^{n} a_{ij} y_j \leq w \ (i = 1, 2, \ldots, m)$$

$$e^T y = 1$$

$$y \geq 0, \tag{4.38}$$

where $w = \underset{1 \leq i \leq m}{Max} \left(\sum_{j=1}^{n} a_{ij} y_j \right)$.

Now it can be verified that (4.37) and (4.38) constitute a primal-dual pair of linear programming problems. Since both maxmin and minmax are attained, these two LPPs have optimal solutions ($\overline{x}$ and $\overline{y}$) and therefore by the linear programming duality, the optimal values of (4.37) and (4.38) will be equal. Let this common value be $\overline{v}$. Then the way (4.37) and (4.38) have been constructed, it is obvious that $\sum_{i=1}^{m} a_{ij} \overline{x_i} \geq \overline{v}$, $(j = 1, 2, \ldots, n)$ and $\sum_{j=1}^{n} a_{ij} \overline{y_j} \leq \overline{v}$, $(i = 1, 2, \ldots, m)$ implying that $(\overline{x})^T A y \geq \overline{v}$ for all $y \in S^n$ and $x^T A \overline{y} \leq \overline{v}$ for all $x \in S^m$.

The above discussion then leads to the following equivalence theorem.

Theorem 4.10.1 *The triplet* $(\overline{x}, \overline{y}, \overline{v}) \in S^m \times S^n \times \mathbf{R}$ *is a solution of the game* G *if and only if* $\overline{x}$ *is optimal to* (4.37), $\overline{y}$ *is optimal to* (4.38) *and* $\overline{v}$ *is the common value of* (4.37) *and its dual* (4.38).

Thus, we have concluded that the matrix game $G = (S^m, S^n, A)$ is equivalent to the primal-dual linear programming problems (4.37)-(4.38).

The pair (4.37)-(4.38) can further be expressed in the form ($LP2$)-($LD2$) where duality is much more obvious and it does not need any checking. For this we need to assume that v^*, the value of the game G, is positive. This assumption can be taken without any loss of generality since matrix games $G = (S^m, S^n, A)$ and $G_1 = (S^m, S^n, A_1)$, $A_1 = (a_{ij} + \alpha)$, $\alpha \in \mathbf{R}$ will have same optimal strategies but different values as v^* and ${v_1}^*$ where ${v_1}^* = v^* + \alpha$. The consequence of the assumption that $v^* > 0$ is that in (4.37) and (4.38) we have $v > 0$ and $w > 0$. Now by defining $x_i' = \frac{x_i}{v}$, $y_j' = \frac{y_j}{w}$ $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ and noting that $e^T x' = \frac{1}{v} e^T x = \frac{1}{v}$ and $e^T y' = \frac{1}{w} e^T y = \frac{1}{w}$, and $Max\ v$ (respectively $Min\ w$) $= Min\left(\frac{1}{v}\right)\Big($ respectively $Max\left(\frac{1}{w}\right)\Big)$, problems (4.37) and (4.38) become

$$\begin{aligned} &\text{Min} && e^T x' \\ &\text{subject to} \\ &&& \sum_{i=1}^{m} a_{ij} x_i' \geq 1 \quad (j = 1, 2, \ldots, n) \\ &&& x' \geq 0, \end{aligned} \tag{4.39}$$

and

$$\begin{aligned} &\text{Max} && e^T y' \\ &\text{subject to} \\ &&& \sum_{j=1}^{n} a_{ij} y_j' \leq 1 \quad (i = 1, 2, \ldots, n) \\ &&& y' \geq 0. \end{aligned} \tag{4.40}$$

Since (4.39)-(4.40) constitutes a primal-dual pair, it is enough to solve only one of these as the solution of the other will be obtained directly because of the duality theory. Once optimal solution $x^{*'}$ of (4.39) and $y^{*'}$ of (4.40) are obtained, the value of the game G is obtained as $v^* = w^* = \frac{1}{e^T x^{*'}} = \frac{1}{e^T y^{*'}}$. Also, optimal strategies for Player I and Player II are obtained as $x^* = v^* x^{*'}$ *and* $y^* = v^* y^{*'}$ respectively.

In the above discussion we have constructed a primal-dual pair (4.37)-(4.38) (or (4.39)-(4.40)) for a given general two person zero-sum matrix game G. It is now natural to ask what happens if we are given any general pair of primal-dual linear programming problems say (4.41) and (4.42). Can we construct an equivalent matrix game G? The answer is in affirmative and that is what we discuss now.

Consider the linear programming problem (4.41) together with its dual (4.42) as follows

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ &&& Ax \leq b \\ &&& x \geq 0, \end{aligned} \tag{4.41}$$

and

$$\begin{aligned} &\text{Min} && b^T y \\ &\text{subject to} \\ &&& A^T y \geq c \\ &&& y \geq 0, \end{aligned} \tag{4.42}$$

where $c \in \mathbf{R}^n$, $x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $y \in \mathbf{R}^m$, $A = (a_{ij})$ is an $(m \times n)$ real matrix.

Now, consider the matrix game associated with the following $(n+m+1)\times(n+m+1)$ skew-symmetric matrix

$$B = \begin{pmatrix} 0 & -A^T & c \\ A & 0 & -b \\ -c^T & b^T & 0 \end{pmatrix}.$$

Since B is a skew-symmetric matrix, the value of the matrix game associated with B is zero and both players have the same optimal strategies. In the following, the matrix game B will mean the matrix game associated with B and indices i and j will run from 1 to m and 1 to n respectively. Also a strategy for either player will be denoted by (x, y, z) where $x \in \mathbf{R}^n$, $y \in \mathbf{R}^m$ and $z \in \mathbf{R}$.

The following result shows that the primal-dual pair (4.41)-(4.42) is equivalent to the matrix game B.

Theorem 4.10.2 *Let $\bar{x}$ and $\bar{y}$ be optimal to* (4.41) *and* (4.42) *respectively. Let* $z^* = \dfrac{1}{1+\sum_j \bar{x}_j + \sum_i \bar{y}_i}$, $x^* = z^*\bar{x}$, $y^* = z^*\bar{y}$. *Then (x^*, y^*, z^*) solves the matrix game B.*

Proof. First we show that $Z^* = (x^*, y^*, z^*)$ will be an optimal strategy for both the players. For this we note that

$$x^* + y^* + z^* = \sum \bar{x}_j z^* + \sum \bar{y}_i z^* + z^* = (1 + \sum \bar{x}_j + \sum \bar{y}_i)z^* = 1,$$

and therefore $(x^*, y^*, z^*) \in S^{m+n+1}$. Now to prove that $Z^* = (x^*, y^*, z^*)$ is an optimal strategy for Player II, we have to show that $BZ^* \le 0$.

But $\bar{x}$ and $\bar{y}$ are solutions of (4.41) and (4.42) and therefore by the duality theory

$$\begin{aligned} c - A^T\bar{y} &\le 0 \\ A\bar{x} - b &\le 0 \\ -c^T\bar{x} + b^T\bar{y} &\le 0. \end{aligned}$$

On multiplying these inequalities by z^*, we have

$$\begin{aligned} cz^* - A^T y^* &\le 0 \\ Ax^* - bz^* &\le 0 \\ -c^T x^* + b^T y^* &\le 0, \end{aligned}$$

which on writing in matrix form gives $BZ^* \le 0$.

Now we note that B is skew symmetric and therefore $BZ^* \le 0$ gives $(Z^*)^T B \ge 0$, which implies that Z^* is an optimal strategy for Player I as well.

Theorem 4.10.3 *Let (x^*, y^*, z^*) be an optimal strategy of the matrix game B with $z^* > 0$. Let* $\bar{x}_j = \dfrac{x_j^*}{z^*}$, $\bar{y}_i = \dfrac{y_i^*}{z^*}$. *Then $\bar{x}$ and $\bar{y}$ are optimal solutions to* (4.41) *and* (4.42) *respectively.*

Proof. Since both players have the same optimal strategies, it is sufficient to take $Z^* = (x^*, y^*, z^*)$ as an optimal strategy for either player, say Player II. Similar arguments are valid if Z^* is taken as an optimal strategy for Player I. Therefore, let $Z^* = (x^*, y^*, z^*)$ be an optimal strategy for Player II with $z^* > 0$. Then we have

$$\begin{aligned} -A^T y^* + cz^* &\leq 0 \\ Ax^* - bz^* &\leq 0 \\ -c^T x^* + b^T y^* &\leq 0. \end{aligned}$$

Now $-A^T y^* + cz^* \leq 0$ gives $A^T\left(\frac{y^*}{z^*}\right) \geq c$. Similarly the other two inequalities gives $A\left(\frac{x^*}{z^*}\right) \leq b$ and $c^T\left(\frac{x^*}{z^*}\right) \geq b^T\left(\frac{y^*}{z^*}\right)$. Therefore we have

$$A^T\bar{y} \geq c,$$

$$A\bar{x} \leq b,$$

and

$$c^T\bar{x} \geq b^T\bar{y}.$$

But the first two inequalities imply

$$c^T\bar{x} \leq \bar{y}^T A\bar{x} \leq \bar{y}^T b = b^T\bar{y},$$

and therefore we have $c^T\bar{x} = b^T\bar{y}$.
This proves that $\bar{x}$ and $\bar{y}$ are optimal for the primal and dual problems respectively. □

Thus the equivalence between two person zero-sum matrix game theory and duality in linear programming is complete in the sense that given any general two person zero sum matrix game G, there is a related pair of primal-dual linear programming problems, and given any general pair of primal-dual linear programming problems, there is an associated matrix game B.

Example 4.10.1 *Solve the following* (3×3) *game where pay-off matrix is*

$$\begin{bmatrix} 3 & -1 & -3 \\ -3 & 3 & -1 \\ -4 & -3 & 3 \end{bmatrix}$$

Solution We shall make use of the Theorem 4.10.1 to get the solution of the given game. For this we need to construct the two linear programming problems (one for Player I and the other for Player II) and solve the same by the simplex method. As per our

discussion of Section 4.10, the LPP for Player I is

$$\begin{aligned} &\text{Max} && v \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} 3x_1 - 3x_2 - 4x_3 &\geq v \\ -x_1 + 3x_2 - 3x_3 &\geq v \\ -3x_1 - x_2 + 3x_3 &\geq v \\ x_1 + x_2 + x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0, \end{aligned} \tag{4.43}$$

while the LPP for Player II is

$$\begin{aligned} &\text{Min} && w \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} 3y_1 - y_2 - 3y_3 &\leq w \\ -3y_1 + 3y_2 - y_3 &\leq w \\ -4y_1 - 3y_2 + 3y_3 &\leq w \\ y_1 + y_2 + y_3 &= 1 \\ y_1, y_2, y_3 &\geq 0. \end{aligned} \tag{4.44}$$

Solving the above LPP's by the simplex method we get

$$x_1^* = 20/45, x_2^* = 11/45, \; x_3^* = 14/45$$

$$y_1^* = 14/45, y_2^* = 11/45, \; x_3^* = 20/45$$

and

$$v^* = w^* = -29/45$$

Therefore the value of the given game is -29/45. Also, $x^* = (20/45, 11/45, 14/45)$ and $y^* = (14/45, 11/45, 20/45)$ are optimal strategies for Player I and Player II respectively.

We can also solve the given game by solving the LPP's (4.39) and (4.40). But for this we need to assume that the value of the game is positive. Since the pay-off matrix has negative elements, to guarantee that the value of the given game is positive, we need to add a suitable positive number to all elements of the pay off matrix and then solve the LPP's (4.39) and (4.40) corresponding to this new pay-off matrix. For our example we may add the number 5 to all elements of the pay-off matrix matrix to get the new pay-off matrix as

$$\begin{bmatrix} 8 & 4 & 2 \\ 2 & 8 & 4 \\ 1 & 2 & 8 \end{bmatrix}$$

Now the LPP's (4.39) and (4.40) corresponding to above pay-off matrix are

$$\begin{aligned} &\text{Min} && x_1' + x_2' + x_3' \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} 8x_1' + 2x_2' + x_3' &\geq 1 \\ 4x_1' + 8x_2' + 2x_3' &\geq 1 \\ 2x_1' + 4x_2' + 8x_3' &\geq 1 \\ x_1', x_2', x_3' &\geq 0, \end{aligned} \tag{4.45}$$

and

$$\begin{aligned} &\text{Max} && y_1' + y_2' + y_3' \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} 8y_1' + 4y_2' + 2y_3' &\leq 1 \\ 2y_1' + 8y_2' + 4y_3' &\leq 1 \\ y_1' + 2y_2' + 8y_3' &\leq 1 \\ y_1', y_2', y_3' &\geq 0, \end{aligned} \tag{4.46}$$

respectively.

On solving the above two LPP's by the simplex method we get

$$x_1' = 5/49, x_2' = 11/196, \; x_3' = 1/14$$

$$y_1' = 1/14, y_2' = 11/196, \; y_3' = 5/49$$

Therefore the value of the game is $v^* = (1/(x_1' + x_2' + x_3') - 5) = (1/(y_1' + y_2' + y_3') - 5) = 196/45 - 5 = -29/45$. Further the optimal strategies for Player I and Player II are

$$x_1^* = 20/45, x_2^* = 11/45, \; x_3^* = 14/45$$

and

$$y_1^* = 14/45, y_2^* = 11/45, \; y_3^* = 20/45$$

respectively. Here for $i, j = 1, 2, 3$ we have $x_i^* = \dfrac{x_i'}{\sum x_i'}$ and $y_j^* = \dfrac{y_i'}{\sum y_i'}$.

Example 4.10.2 *Obtain the matrix game corresponding to the primal-dual pair for the LPP*

$$\begin{aligned} &\textit{Max} && 4x_1 + 3x_2 \\ &\textit{subject to} \end{aligned}$$

$$\begin{aligned} x_1 + x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 10 \\ x_1, x_2 &\geq 0\,, \end{aligned} \tag{4.47}$$

and also solve the game so obtained.

Solution The dual of the given LPP is

$$\begin{aligned} &\text{Min} && 8x_1 + 10x_2 \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} w_1 + 2w_2 &\geq 4 \\ w_1 + w_2 &\leq 3 \\ w_1, w_2 &\geq 0. \end{aligned} \tag{4.48}$$

As $c = (4,3)^T$, $b = (8,10)^T$ and $A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$, the matrix game corresponding to the given primal-dual pair of LPP's is

$$B = \begin{bmatrix} 0 & 0 & -1 & -2 & 4 \\ 0 & 0 & -1 & -1 & 3 \\ 1 & 1 & 0 & 0 & -8 \\ 2 & 1 & 0 & 0 & 10 \\ -4 & -3 & 8 & 10 & 0 \end{bmatrix}$$

Let us recall that the optimal solution of the given LPP is $(x_1^* = 2, x_2^* = 6)$, and that of its dual is $(w_1^* = 2, w_2^* = 1)$. Therefore we can use Theorem 4.10.2, to obtain the solution of the game B. We note that $z^* = 1/(1+8+3) = 1/12$. Therefore optimal strategy for Player I and also for Player II is $(x_i' = zx_i^*, y_j' = zy_j^*)$ for $i = 1,2$ and $j = 1,2$, i.e. $(x_1' = 1/6$, $x_2' = 1/2$, $y_1' = 1/6$, $y_2' = 1/12$, $z' = 1/12)$. Further the value of the game is zero.

4.11 Summary and Additional Notes

- This chapter pertains to the study of duality theory in linear programming. Apart from proving various duality theorems in Sections 4.2-4.5, a separate section, namely Section 4.6, is devoted to understand certain useful deductions of the complementary slackness theorem. It is shown that if the parameters of given LPP have certain economic meaning then the dual can also be given a meaningful economic interpretation.

- Section 4.7 presents the dual simplex method which is then used to study the post optimality analysis. Here it is also remarked that though the dual simplex method may not be of significant use in solving LPP's, it has a very important role to play in the study of integer linear programming problems.
- Section 4.9 introduces two person zero sum matrix games and establishes an equivalence between such a game and a related primal-dual (pair) of LPP's. This equivalence provides a solution procedure for solving such matrix games by the simplex method.
- Some of the standard texts on duality are Murty [117], Bazaraa et al. [12], Gass [66] and Hadley [72]. For the equivalence between a two person zero-sum matrix game and LP duality, the texts by Owen [120], Karlin [90], and Parthasarathy and Raghvan [128] are appropriate references.
- The basic idea of duality in LP is due to John Von Neumann, (Von Neumann and Morgenstern [118]) in the context his work in game theory. This idea was later developed by D. Gale, H. W. Kuhn and A. W. Tucker in 1951, Tucker in 1960, and many other researchers.
- The dual simplex method was first developed by C.E. Lemke in 1954. S. Zionts in 1969 developed the cris cross algorithm which combines the primal and dual simplex methods in an interesting manner.
- To understand the economic interpretations of various duality results in linear programming, we may refer to Gale [63], Samuelson [137], and Intriligator [80].

4.12 Exercises

4.1 *Write the dual of the following LPP's*

(1) Max $5x_1 + 2x_2 - x_3$

subject to

$$x_1 + 5x_2 - 2x_3 = 0$$
$$x_1 - 5x_2 - 6x_3 \geq 40$$
$$x_2 + 2x_3 \geq 13$$
$$x_1 \text{ unrestricted}, x_2 \leq 0, x_3 \geq 0 .$$

(2) Min $2x_1 + 3x_2 + 4x_3$

subject to

$$2x_1 + 2x_2 + 3x_3 \leq 4$$
$$3x_1 + 4x_2 + 5x_3 \geq 5$$
$$x_1 + 2x_2 + x_3 = 7$$
$$x_1 \geq 0, x_2 \text{ unrestricted}, x_3 \geq 0 .$$

(3) *Max* $x_1 - x_2 + x_3 - x_4$

subject to

$$0 \le x_1 \le 8$$
$$-4 \le x_2 \le 4$$
$$-2 \le x_3 \le 4$$
$$0 \le x_4 \le 10.$$

(4) *Min* $x_1 + x_2$

subject to

$$x_1 + 2x_2 \ge 6$$
$$7x_1 + 8x_2 \le 56$$
$$x_1 \ge 0, x_2 \le 0.$$

4.2 *Write the dual* (D) *of the following LPP* (P)

Max $5x_1 + 2x_2 + 3x_3$

subject to

$$x_1 + 5x_2 + 2x_3 = 30$$
$$x_1 - 5x_2 - 6x_3 \le 40$$
$$x_1, x_2, x_3 \ge 0.$$

Solve (D) *by solving* (P) *and also verify the complementary slackness theorem.*

4.3 *The initial simplex tableau of a LPP in the maximization form (no artificial variables are required) is as follows*

	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$
$x_5 = 6$	4	9	7	10	1	0
$x_6 = 4$	1	1	3	40	0	1
0	−12	−20	−18	−40	0	0

1. Write (*P*) and (*D*).
2. Solve (*P*) by the simplex method and hence find the optimal solution of the dual problem.
3. Verify the complementary slackness theorem.

4.4 *Write the dual of*

Max $x_1 - x_2 + x_3 - x_4$

subject to

$$x_1 + x_2 + x_3 + x_4 \le 10$$
$$0 \le x_1 \le 8$$
$$-4 \le x_2 \le 4$$
$$-1 \le x_3 \le 4$$
$$0 \le x_4 \le 10,$$

and show that $x_1^* = 8,\ x_2^* = -4,\ x_3^* = 4,\ x_4^* = 0$ *is an optimal solution of the given LPP. (Hint: Use the complementary slackness theorem).*

4.5 *Use the complementary slackness theorem to verify that* $(n, 0, \ldots, 0)$ *is an optimal solution of the LPP*

$$\text{Min} \quad \sum_{j=1}^{n} j\, x_j$$

subject to

$$\sum_{j=i}^{n} x_j \geq i \ (i = 1, 2, \ldots, n)$$

$$x_j \geq 0 \ (j = 1, 2, \ldots, n).$$

4.6 *Write the dual of the following LPP*

$$\text{Min} \quad 5x_1 + 4x_2 + 6x_3 + 6x_4 + 3x_5 + 2x_6$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 11 \\ x_4 + x_5 + x_6 &= 16 \\ x_1 + x_4 &= 9 \\ x_2 + x_5 &= 5 \\ x_3 + x_6 &= 13 \\ x_j \geq 0, \ (j &= 1, 2, 3, 4, 5, 6). \end{aligned}$$

4.7 *Consider the LPP* (P)

$$\text{Max} \quad 2x_1 + 5x_2 + 3x_3$$

subject to

$$\begin{aligned} 4x_1 + x_2 &\leq 420 \\ 2x_2 + 3x_3 &\leq 460 \\ 2x_1 + x_2 + x_3 &\leq 500 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

1. *Write the dual* (D) *of* (P).
2. *It is claimed that the optimal value of* (P) *is in the interval* $[1150, 2500]$. *Verify the claim without carrying out any simplex computation on either* (P) *or* (D). *Let it be known that* $(0, \alpha, 0)$ *and* $(0, 0, \beta)$ *is feasible for* (P) *and* (D) *respectively for some* α *and* β. *State the duality theorem used in this regard.*

4.8 *It is claimed that the dual* (D) *of the following LPP* (P) *has no optimal solution, where* (P) *is*

$$\begin{aligned} \text{Max} \quad & 3x_1 - 4x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 4. \end{aligned}$$

Use the simplex algorithm to verify this claim without writing or solving (D). *State the duality theorem to be used in this regard.*

4.9 *Consider the following LPP* (P)

$$\text{Min} \quad 2x_1 + 15x_2 + 5x_3 + 6x_4$$

subject to

$$\begin{aligned} x_1 + 6x_2 + 3x_3 + x_4 &\geq 2 \\ 2x_1 - 5x_2 + x_3 - 3x_4 &\geq 3 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

1. *Write the dual* (D) *of the above LPP* (P).
2. *Solve* (D) *graphically.*
3. *Utilizing the information in (ii) above and various theorems of duality, obtain an optimal solution of* (P).

4.10 *Solve the following LPP by the dual simplex method*

$$\text{Min} \quad x_1 + 2x_2 + 3x_3 + 4x_4$$

subject to

$$\begin{aligned} x_1 + 2x_2 + 2x_3 + 3x_4 &\geq 30 \\ 2x_1 + x_2 + 3x_3 + 2x_4 &\geq 20 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

4.11 *Consider the LPP*

$$\text{Max} \quad 5x_1 + 12x_2 + 4x_3$$

subject to

$$\begin{aligned} x_1 + 2x_2 + x_3 &\leq 10 \\ 2x_1 - x_2 + 3x_3 &= 8 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Without using the simplex algorithm, find the optimal solution of the dual, given that x_1, x_2 *are strictly positive in the optimal solution.*

4.12 *Using simplex method, solve the following linear programming problem:*

$$\text{Max} \quad 3x_1 + 2x_2$$

subject to

$$\begin{aligned} x_1 + 2x_2 &\leq 6 \\ 2x_1 + x_2 &\leq 6 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Append an additional constraint $x_1 + x_2 \geq 3$ *in the optimal simplex tableau. Does the original optimal solution remain optimal?*

Update the original optimal simplex tableau if an additional variable x_3 *with activity vector* $a^{(3)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ *and price* $c_3 = 2$ *is introduced in the original problem and find its optimal solution.*

4.13 *Consider the linear programming problem*

$$\begin{aligned} &\textit{Max} && x_1 + 6x_2 \\ &\textit{subject to} \\ & && x_1 + 3x_2 \le 12 \\ & && 3x_1 + x_2 \le 12 \\ & && x_1 + x_2 = 13 \\ & && x_1, x_2 \ge 0, \end{aligned}$$

and let following be its last tableau

	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
$x_2 = 3$	0	1	3/8	−1/8	0
$x_1 = 3$	1	0	1/8	3/8	0
$x_7 = 7$	0	0	1/4	−3/4	1

1. *What conclusion can you draw from the tableau?*
2. *Update the table if b is changed to $(12, 12, 6)^T$, what do you infer now?*
3. *Identify the redundant constraint (if any) and hence find the new optimal basic feasible solution of the reduced problem?*
4. *Does the addition of the constraint $x_1 + x_2 = 3$ in the reduced problem affects the solution. If yes, then find the optimal basic feasible solution.*

4.14 *Consider the following LPP problem and its optimal tableau:*

$$\begin{aligned} &\textit{Max} && 2x_1 + x_2 - x_3 \\ &\textit{subject to} \\ & && x_1 + 2x_2 + x_3 \le 8 \\ & && -x_1 + x_2 - 2x_3 \le 4 \\ & && x_1, x_2, x_3 \ge 0. \end{aligned}$$

	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$
$x_1 = 8$	1	2	1	1	0
$x_5 = 12$	0	3	−1	1	1
16	0	3	3	2	0

1. *Write the dual problem and find the optimal dual solution from the optimal tableau.*
2. *Find new optimal solution if the coefficient of x_2 in the objective function is changed from 1 to 6.*
3. *Find new optimal solution if the coefficient of x_2 in the first constraint is changed from 2 to $\frac{1}{4}$.*
4. *What will be the change in the optimal solution, if a new constraint $x_2 + x_3 = 3$ is added?*
5. *What will be the change in the optimal solution, if a new variable x_4 is introduced with cost vector 4 and consumption vector $(1, 2)^T$?*

4.15 *Show that the LPP*

$$\begin{aligned} &\textit{Max} && c^T x \\ &\textit{subject to} \\ & && Ax = b \\ & && x \in \mathbf{R}^n , \end{aligned}$$

has an optimal solution if and only if c is a linear combination of columns of A.

4.16 *Let the LPP*

$$\begin{aligned} &\textit{Min} && c^T x \\ &\textit{subject to} \\ & && Ax = b \\ & && x \geq 0 , \end{aligned}$$

have a finite optimal solution. Using duality theory show that the LPP

$$\begin{aligned} &\textit{Min} && c^T x \\ &\textit{subject to} \\ & && Ax = \bar{b} \\ & && x \geq 0 , \end{aligned}$$

cannot be unbounded, no matter what value the vector $\bar{b}$ might take.

4.17 *Prove that LPP's*

$$\begin{aligned} &\textit{Max} && c^T x \\ &\textit{subject to} \\ & && Ax \leq b \\ & && x \geq 0 , \end{aligned}$$

and

$$\begin{aligned} &\textit{Min} && b^T w \\ &\textit{subject to} \\ & && A^T w \geq c \\ & && w \geq 0 , \end{aligned}$$

have optimal solutions if and only if the system

$$\begin{aligned} Ax + u &= b \\ A^T w - v &= c \\ c^T x - b^T w &= 0 \\ x \geq 0, w \geq 0, u \geq 0, v &\geq 0 , \end{aligned}$$

is consistent.

4.18 *Are the following statements true? Give reasons for your answer*

1. *The primal LP (P) and its dual LP (D), both cannot have unbounded solution.*
2. *The primal LP (P) and its dual LP (D), both cannot be infeasible.*
3. *The primal-dual pair of LP's can be solved by solving a system of linear equations for non-negative variables.*
4. *The dual(dual(dual)) of a LPP is the primal LPP.*
5. *If the primal LP (P) has unique optimal solution and the dual LP (D) is feasible, then (D) also has unique optimal solution.*

4.19 *Consider the LPP*

$$\begin{aligned} &Max \quad (c^T x - b^T w) \\ &\text{subject to} \\ &\qquad Ax \le b \\ &\qquad A^T w \ge c \\ &\qquad x,\ w \ge 0\,. \end{aligned}$$

Show that the above LPP is either infeasible or its optimal value is zero.

4.20 *Show that the system $A^T w = c,\ w \ge 0$ is inconsistent if and only if the system $Ax \le 0,\ c^T x > 0$ is consistent.*

4.21 *Use linear programming problem to solve the matrix game*

$$\begin{bmatrix} 1 & 3 & -3 & 7 \\ 2 & 5 & 4 & -6 \end{bmatrix}$$

4.22 *Solve the following game by the simplex method*

$$\begin{bmatrix} 1 & 2 \\ 5 & 6 \\ 2 & 1 \end{bmatrix}$$

4.23 *Solve the game corresponding to the primal-dual pair of LPP's where the given LPP is*

$$\begin{aligned} &Min \qquad x_1 + 2x_2 \\ &\text{subject to} \\ &\qquad 4x_1 + 2x_2 \le 4 \\ &\qquad -3x_1 + 3x_2 \ge 2 \\ &\qquad x_1, x_2 \ge 0. \end{aligned}$$

5
The Transportation and Assignment Problems

5.1 Introduction

In this chapter we study two special linear programming problems which have traditionally been applied in many real life situations. These problems are the classical *transportation and assignment problems*. One key feature of these problems is that, in general, they require a very large number of constraints and variables and so a straight forward application of the simplex method will turn out to be highly inefficient. However, the coefficient matrices of both the transportation and the assignment problems have a very special structure. Because of this specific structure, it is possible to develop a very simplified version of the simplex method for solving these problems that achieves dramatic computational savings in its implementation. We discuss these details in the subsequent sections.

5.2 The Transportation Problem : Description and Mathematical Model

In this section we first describe the single commodity cost minimizing transportation problem and then obtain a mathematical model of the same. For this, let us assume that a single commodity is available at each of the m sources (stores) in $a_1, a_2, \ldots, a_m$ units respectively. This commodity is required at each of the n destinations (shops) in $b_1, b_2, \ldots, b_n$ units respectively. Therefore the commodity has to be transported from sources to destinations. Let c_{ij} $(i = 1, 2, \ldots, m\ ;\ j = 1, 2, \ldots, n)$ be the per unit cost of transportation of the commodity from the ith source to the jth destination. In the cost minimizing transportation problem, our aim is to find how many units of the given commodity should be transported from each source to each destination so that the total cost of transportation is minimum.

To get a mathematical model of the above problem we have to first decide the total number of decision variables. Since there are m sources and n destinations and we have to find the units of commodity to be transported from each source to each destination, there are mn decision variables, x_{ij}. Here x_{ij} denotes the units of commodity to be transported from the ith source to the jth destination $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$.

Now we have to express mathematically the physical and technological restrictions on the problem in terms of the parameters and decision variables. For the transportation problems there are three types of physical constraints. These are (i) from any source, we should not transport more than what is available (ii) to every destination, we should transport at least what is required and (iii) we can not transport negative units of commodity.

Let us now consider a typical source i. The total quantity leaving the ith source is $x_{i1} + x_{i2} \ldots x_{in}$. As the maximum availability of the commodity at the ith source is a_i units, we should have $x_{i1} + x_{i2} + \ldots + x_{in} \leq a_i$ and this has to hold for each $i = 1, 2, \ldots, m$. Thus,

$$\sum_{j=1}^{n} x_{ij} \leq a_i \quad (i = 1, 2, \ldots, m). \tag{5.1}$$

The m constraints given at (5.1) are called the *source constraints.*

Similarly if we consider a typical destination j, then the total quantity reaching at destination j is $x_{1j} + x_{2j} + \ldots + x_{mj}$. As the requirement at the destination j is at least b_j units, we should have $x_{1j} + x_{2j} + \ldots + x_{mj} \geq b_j$, and this has to hold for each $j = 1, 2, \ldots, n$. Thus,

$$\sum_{i=1}^{m} x_{ij} \geq b_j \quad (j = 1, 2, \ldots, n). \tag{5.2}$$

The n constraints given at (5.2) are called *destination constraints.*

Further, as we can not transport negative units of commodity we have

$$x_{ij} \geq 0 \quad (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n). \tag{5.3}$$

We next consider the quantity to be optimized, which in our case is the total cost of transportation. As c_{ij} denotes the per unit cost of transporting the commodity from the ith source to the jth destination, under the assumptions of proportionality and additivity, i.e. linearity, the total cost of transportation is

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{5.4}$$

Combining (5.1), (5.2), (5.3) and (5.4), we get the following as the mathematical model of the transportation problem (TP)

$$\text{Min} \quad z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le a_i \quad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j \quad (j = 1,2,\ldots,n)$$

$$x_{ij} \ge 0 \quad (i = 1,2,\ldots,m; j = 1,2,\ldots,n). \tag{5.5}$$

Looking at problem (5.5) above, we observe that here the objective function as well as all constraints are linear functions of decision variables x_{ij} so this is a linear programming problem and therefore can certainly be solved by the usual simplex method. However, as we shall see in the next section, it is a very special type of LPP where the coefficient matrix A has a very specific structure. Therefore we expect that the matrix A may have certain nice properties which can possibly be exploited so that the implementation of the simplex method for the transportation problem is much more simple and efficient. This is really true and it will be clear as we proceed with our discussion.

5.3 The Balanced Transportation Problem

The transportation problem (5.5) is called the *balanced transportation problem* (BTP) if all the source and destination constraints hold as equations, i.e.

$$\text{Min} \quad z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = a_i \quad (i = 1,2,\ldots,m),$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1,2,\ldots,n),$$

$$x_{ij} \ge 0 \quad (i = 1,2,\ldots,m; j = 1,2,\ldots,n). \tag{5.6}$$

From (5.6), it is obvious that if it is feasible then total quantity available should be equal to the total quantity required, i.e. $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. What we will like to emphasize here is that the converse is also true, i.e. if $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = \alpha$, then problem (5.6) is feasible. We can verify the same by taking $x_{ij} = (a_i b_j)/\alpha$ for all i and

j. Here it is of course assumed that $a_i \geq 0$, $b_j \geq 0$ for all i and j. In view of the above, we can call a transportation problem to be balanced if and only if $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$; otherwise it is called an *unbalanced transportation problem*.

We shall first present an algorithm for solving the balanced transportation problems only. Later we shall show that if the problem is not balanced then we can make it balanced, and then solve the same.

As remarked earlier, every transportation problem and therefore, in particular, the balanced one i.e. problem (5.6) is a linear programming problem. Let us identify the cost vector c, the resource vector b, the decision vector x and the coefficient matrix A for problem (5.6). It is obvious that we should take

$$\begin{aligned} c &= col\,(c_{11}, \ldots, c_{1n}, \ldots, c_{m1}, \ldots, c_{mn}) \in \mathbf{R}^{m \times n}, \\ x &= col\,(x_{11}, \ldots, x_{1n}, \ldots, x_{m1}, \ldots, x_{mn}) \in \mathbf{R}^{m \times n}, \\ b &= (a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n), \end{aligned}$$

so that the objective function becomes $c^T x$ and constraints become $Ax = b$, $x \geq 0$ where A is the coefficient matrix for the constraints in (5.6). To have the full description of the coefficient matrix A, we write each of the constraints of problem (5.6) in detail, i.e.

$$\begin{aligned} x_{11} + x_{12} + \ldots + x_{1n} &= a_1 \\ x_{21} + x_{22} + \ldots + x_{2n} &= a_2 \\ &\ldots \\ x_{m1} + x_{m2} + \ldots + x_{mn} &= a_m \\ x_{11} + x_{21} + \ldots + x_{m1} &= b_1 \\ x_{12} + x_{22} + \ldots + x_{m2} &= b_2 \\ &\ldots \\ x_{1n} + x_{2n} + \ldots + x_{mn} &= b_m. \end{aligned}$$

Here the first m constraints are the source constraints and the next n constraints are the destination constraints. Since x and b have already been identified, the above gives

$$A = \begin{bmatrix} 1\,1\ldots\ldots 1\,0\,0\ldots\ldots 0\ldots & 0\,0\ldots\ldots 0 \\ 0\,0\ldots\ldots 0\,1\,1\ldots\ldots 1\ldots & 0\,0\ldots\ldots 0 \\ \ldots\ldots\ldots\ldots & \ldots \\ 0\,0\ldots\ldots 0\,0\,0\ldots\ldots 0\ldots & 1\,1\ldots\ldots 1 \\ 1\,0\ldots\ldots 0\,1\,0\ldots\ldots 0\ldots & 1\,0\ldots\ldots 0 \\ 0\,1\ldots\ldots 0\,0\,1\ldots\ldots 0\ldots & 0\,1\ldots\ldots 0 \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots & \vdots\ \vdots\ \vdots \\ 0\,0\ldots\ldots 1\,0\,0\ldots\ldots 1\ldots & 0\,0\ldots\ldots 1 \end{bmatrix}, \tag{5.7}$$

which is the $(m+n) \times mn$ coefficient matrix.

We see here that all entries in A are 0 and 1 only and they appear in certain pattern. As the first m rows of A came from the source constraints, we call them *source rows*

and denote them by $s^{(i)}$, $(i = 1, 2, \ldots, m)$. Similarly, the last n rows of A came from the destination constraints, so we call them *destination rows* and denote them by $d^{(j)}$, $(j = 1, 2, \ldots, n)$. Further in each of the mn columns of A there are two 'ones', one 'one' coming because of the source rows and the other 'one' coming because of the destination rows. Due to this specific structure of A, the following properties are possessed by the matrix A which we state in the form of following lemmas.

Lemma 5.3.1 *Rank* $A < (m + n)$.

Proof. As neither $m = 1$ nor $n = 1$, $(m + n)$ is always less than mn. So we shall prove that Rank $A \neq (m + n)$. This follows clearly because

$$\sum_{i=1}^{m} s^{(i)} = (1, 1, \ldots, 1) \in \mathbf{R}^{m \times n}$$

and

$$\sum_{j=1}^{n} d^{(j)} = (1, 1, \ldots, 1) \in \mathbf{R}^{m \times n},$$

which gives

$$\sum_{i=1}^{m} s^{(i)} - \sum_{j=1}^{n} d^{(j)} = 0. \tag{5.8}$$

But (5.8) means that the rows of A are linearly dependent which proves that Rank $A < (m + n)$. □

Lemma 5.3.2 *Rank*$A = (m + n - 1)$

Proof. The proof follows directly from (5.8) because on the left hand side, all coefficients are non-zero, so any row can be written in terms of the remaining $(m + n - 1)$ rows. In fact, this also shows that if we delete any row from A, the remaining $(m + n - 1)$ rows are linearly independent.

We can also give a more explicit proof by actually constructing an $(m+n-1)\times(m+n-1)$ submatrix of A whose determinant is not zero. For this we first construct the submatrix $F = \{n^{th}column,\ 2n^{th}column, \ldots, mn^{th}$column, $1^{st}column, \ldots, (n-1)^{th}column\}$ which is of order $(m + n) \times (m + n - 1)$ as every column in A has $(m + n)$ elements. Now we delete the $(m + n)^{th}$ row from F to get an $(m + n - 1) \times (m + n - 1)$ submatrix,

$$D = \begin{pmatrix} 1\,0\ldots0 \vdots 1\,1\ldots1 \\ 0\,1\ldots0 \vdots 0\,0\ldots0 \\ \ldots\ldots\vdots\ldots\ldots \\ 0\,0\ldots1 \vdots 0\,0\ldots0 \\ \ldots\ldots\ldots\ldots\ldots \\ 0\,0\ldots0 \vdots 1\,0\ldots0 \\ 0\,0\ldots0 \vdots 0\,1\ldots0 \\ \ldots\ldots\vdots\ldots\ldots \\ 0\,0\ldots0 \vdots 0\,0\ldots1 \end{pmatrix} = \begin{pmatrix} I_m & \vdots & H \\ \ldots\ldots & & \ldots \\ 0 & \vdots & I_{n-1} \end{pmatrix} \tag{5.9}$$

where, I_k denotes the $(k \times k)$ identity matrix. Therefore (5.9) gives
$|D| = |I_m||I_{n-1}| = 1 \neq 0$ □

Definition 5.3.1 (Unimodular Matrix). *Let M be a matrix whose entries are $0,1$ and -1 only. Then the matrix M is said to be unimodular if the determinant of every square submatrix of M is either $+1, -1$ or 0.*

Remark 5.3.1 *If the entries in M are $0,1$ and -1, it does NOT follow that M is unimodular. For this we can take the (3×3) matrix*

$$M = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \end{pmatrix}$$

and note that $|M| = -2$. So it is not just the entries (they in any case have to be 0, +1 or -1), but the way they appear in M, i.e their pattern is important.

Lemma 5.3.3 *A is unimodular*

Proof. Let A_k be any $(k \times k)$ submatrix of A. Looking at the structure of the matrix A we note that for A_k exactly one of the following holds

1. there is a column in A_k having all entries as 'zero'.
2. every column of A_k has exactly two 'ones'.
3. there is a column in A_k which has exactly one 'one'.

For the case(i), clearly $|A_k| = 0$, while for the case (ii), one 'one' has to come from a source row and the another 'one' has to come from a destination row. Now similar to Lemma (5.3.1), if we sum over the source rows of A_k and sum over the destination rows of A_k, and then subtract, we get a zero vector. Therefore the rows of A_k are linearly dependent and thus $|A_k| = 0$.

We next consider the case (iii), i.e. there is a column in A_k which has exactly one 'one'. We can therefore open the determinant of A_k and get $|A_k| = \pm|A_{k-1}|$, depending upon the location of this 'one'. Now, the same arguments hold for A_{k-1} as well and this gives either $|A_{k-1}| = 0$ or $|A_{k-1}| = \pm|A_{k-2}|$. Continuing in this manner, we get $|A_k| = 0, +1,$ or -1 as all entries in A are 0 and 1 only. □

5.4 Consequence of Unimodularity

As the rank of the matrix A is $(m+n-1)$, there will be $(m+n-1)$ basic columns. Here it is convenient to take a double suffix notation for the columns of A, i.e. by p_{ij} we denote the $((i-1)n+j)^{th}$ column of A. Thus the first column of A is denoted by p_{11}, second by p_{12} and so on, the $(mn)^{th}$ column being denoted by p_{mn}. Infact, we may consider an $(m \times n)$ *symbolic matrix* having $(m \times n)$ cells, each cell representing a column of A. So, in general, the $(i,j)^{th}$ cell in the symbolic matrix will be identified by the column p_{ij} which is really $((i-1)n+j)^{th}$ column of A. If we further denote by e_i as a vector in $\mathbf{R}^{m \times n}$ having all entries as 'zero' except one entry being 'one' at the ith place, then

$$p_{ij} = e_i + e_{m+j}, \tag{5.10}$$

because every column of A has exactly two 'ones', one 'one' at the ith place and the other 'one' at the $(m+j)^{th}$ place.

Now let $\{p_{\alpha\beta}^{(B)}\}$ denote the $(m+n-1)$ basic columns of A. Let p_{ij} be any other column of A. Then p_{ij} can always be expressed as a linear combination of basic columns, i.e. symbolically

$$p_{ij} = \sum_{(\alpha,\beta)} y_{ij}^{(\alpha,\beta)} p_{\alpha\beta}^{(B)}, \tag{5.11}$$

where the scalars $y_{ij}{}^{(\alpha,\beta)}$ are exactly same as the usual y_{ij} in the simplex algorithm.

Now because of the fact that the coefficient matrix A is unimodular it follows (as we prove below) that the scalars $y_{ij}{}^{(\alpha,\beta)}$ are $+1, -1$ or 0. Also there is a very simple way to exactly find out that which $y_{ij}{}^{(\alpha,\beta)}$ is $+1$, which is -1 and which is zero. Here the symbolic matrix becomes very handy because it allows the whole implementation in an $(m \times n)$ *transportation tableau* only.

If we now recall the following formula from the simplex method

$$\hat{x}_{B_i} = \begin{cases} x_{B_i} - \dfrac{x_{B_r}}{y_{rj}} y_{ij}, & (i \neq r) \\ \dfrac{x_{B_r}}{y_{rj}}, & (i = r), \end{cases} \tag{5.12}$$

and identify $y_{ij}{}^{(\alpha,\beta)}$ by y_{ij} etc, then $y_{rj} > 0$ means $y_{rj} = 1$ because for the transportation problem every scalar $y_{ij} = +1, -1$ or zero. Therefore, for the special case of the transportation problem

$$\hat{x}_{B_i} = \begin{cases} x_{B_i} + x_{B_r}, & y_{ij} = -1 \\ x_{B_i} - x_{B_r}, & y_{ij} = +1 \\ x_{B_i} \quad , & y_{ij} = 0 \end{cases} \tag{5.13}$$

for $i \neq r$ and $\hat{x}_{B_r} = x_{B_r}$.

Looking at (5.13) above, we make an important observation. Because for the transportation problem the coefficient matrix A is unimodular, every coefficient $y_{ij}^{(\alpha,\beta)}$ is $+1$, -1 or zero(see Lemma 1.4.1 below). As a consequence, there is no division/multiplication in the whole algorithm, it involves only the operations of addition, subtraction and comparison. This makes the implementation of the simplex algorithm for the transportation problem extremely simple and efficient.

Lemma 5.4.1 *Let $\{p_{\alpha\beta}{}^{(B)}\}$ be a set of $(m+n-1)$ basic columns of A. Let p_{ij} be any other column of A. Then*

$$p_{ij} = \sum_{(\alpha,\beta)} y_{ij}^{(\alpha,\beta)} p_{(\alpha,\beta)}{}^{(B)}, \tag{5.14}$$

where each $y_{ij}^{(\alpha,\beta)} = +1, -1,$ or zero.

Proof. It is obvious that p_{ij} can always be written as a linear combination of basic columns. So we shall only prove that $y_{ij}^{(\alpha,\beta)} = +1, -1,$ or zero. Let Y_{ij} be a vector whose $(m+n-1)$ entries are $y_{ij}^{(\alpha,\beta)}$ and L be the matrix whose columns are $p_{\alpha,\beta}^{(B)}$. Then equation (5.14) can be written as

$$p_{ij} = LY_{ij}\,. \tag{5.15}$$

But in (5.15) there are $(m+n)$ equations in $(m+n-1)$ unknowns because $p_{ij} \in \mathbf{R}^{m+n}$, $L \in \mathbf{R}^{(m+n)\times(m+n-1)}$ and $Y_{ij} \in \mathbf{R}^{(m+n-1)}$. Therefore, we delete one of the rows from (5.15) say in particular the ith row. Then (5.15) becomes

$$e_{m+j-1} = TY_{ij} \tag{5.16}$$

where T is the matrix obtained from L by deleting the ith row and as $p_{ij} = e_i + e_{m+j}$, after i^{th} row is deleted from p_{ij}, the right hand side of (5.17) becomes e_{m+j-1}. Also $T \in \mathbf{R}^{(m+n-1)\times(m+n-1)}$ is the matrix of basic columns (after deleting the ith row from L), T^{-1} exists. Therefore from (5.16) we get

$$Y_{ij} = T^{-1} e_{m+j-1},$$

i.e,

$$Y_{ij} = (m+j-1)^{th} \text{ column of } T^{-1}. \tag{5.17}$$

But every entry of T^{-1} is the ratio of some cofactor of T and $|T|$. As the matrix A is unimodular (because of Lemma (5.3.3)), $|T|$ as well as all cofactors of T are $+1$, -1 or

zero. But T is invertible, so $|T| = +1$ or -1. Therefore each of these ratios i.e. cofactor of $T/|T|$, are $+1$, -1 or 0. This proves that all entries in T^{-1} are $+1$, -1 or zero. So $Y_{ij} = +1$, -1 or zero. This proves the lemma. □

Now from the application point of view, it is important that we are able to determine the entries of the vector Y_{ij}. Although it is known that all entries of Y_{ij} are $+1$ or -1 or zero, we still do not know how to determine them. In the simplex method the scalars y_{ij} are obtained by actually evaluating $B^{-1}a^{(j)}$, but in the case of the transportation problem the method is extremely simple. We first illustrate the method below for Example 5.4.1 and then give its justification later. Here the role of the symbolic matrix is really important.

Example 5.4.1 *Consider a balanced transportation problem with 2 sources and 4 destinations. Determine the coefficient matrix A and find the expression of a nonbasic column in terms of basic columns.*

Solution Here $m = 2$ and $n = 4$. Therefore the coefficient matrix A will be of order (6×8) as given below

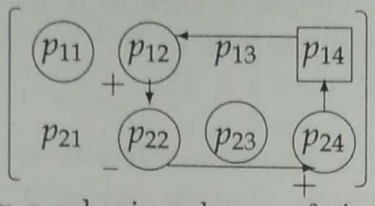
$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The first two rows of A are the source rows and the last four rows of A are the destination row. Also the rank of A is 2+4-1=5.

Now as per our notations, the eight columns of A are denoted by $p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}$, and p_{24} respectively. This gives the symbolic matrix as

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix}$$

Let it be known that the columns $p_{11}, p_{12}, p_{22}, p_{23}$ and p_{24} be five linearly independent columns of A, i.e. these are $(m+n-1)$ basic columns. For the time being, we accept this as we have not yet learnt the method to find initial basic feasible solution for the given balanced transportation problem. The $(m+n-1)$ basic columns of A are identified as $(m+n-1)$ basic cells in the symbolic matrix by encircling them as shown below



Let p_{14} be the given nonbasic column of A, which has to be expressed as a linear combination of the chosen $(m + n - 1)$ basic columns. For this we first identify the corresponding cell (1,4) in the symbolic matrix and enclose it by a rectangle as shown above. Now from the cell (1,4) we start moving in the row so that we get a basic cell such that there is another basic cell in that column. In our example, from the cell (1,4) we come to the basic cell (1,2) as there is a cell (2,2) in that column. Now we start moving from the cell (1,2) in that column so that either we get a basic cell so that there is another basic cell in that row, or we complete the loop. In our example, from the basic cell (1,2) we move to the basic cell (2,2) because there is a basic cell in that row. Now from the basic cell (2,2) we start moving in the row to get the cell (2,4); and then from (2,4) start moving in the column to get the cell (1,4) so as to complete the loop. It is obvious that rather than start moving in the row from the cell (1,4), we can start moving in the column to get the cell (2,4) and then the cells (2,2) and (1,2) respectively.

At this stage we note that for a given nonbasic cell, there certainly exists a loop which is unique. Also the number of basic cells in this loop is always odd. Once the loop for the given nonbasic cell has been obtained we start attaching '+' and '-' sign alternately to every basic cell in the loop. As the number of basic cells in the loop is odd, it does not really matter from which direction we really start. For our example, the loop for the nonbasic cell (1,4) is shown in the symbolic matrix. From the loop we read the expression for p_{14} as $p_{14} = p_{12} - p_{22} + p_{24}$. We can verify that this representation is correct because $p_{14} = e_1 + e_{m+4}$ and $p_{12} - p_{22} + p_{24} = e_1 + e_{m+2} - e_2 - e_{m+2} + e_2 + e_{m+4} = e_1 + e_{m+4}$. Because of the cancelation process involved here, it is obvious that the number of basic cells in the loop for the cell (1,4) must be odd.

In a similar manner we can also obtain the representation of nonbasic columns p_{13} and p_{21} as well, which is

$$p_{13} = p_{12} - p_{22} + p_{23},$$

and

$$p_{21} = p_{22} - p_{12} + p_{11}.$$

Definition 5.4.1 (loop). *An ordered sequence of four or more different cells in the symbolic matrix of the given balanced transportation problem will be called a loop if (i) any two consecutive cells lie in the same row or in the same column, and (ii) no three consecutive cells lie in the same row or in the same column, where the first cell is considered to follow the last cell in the sequence.*

Symbolically, a typical loop will look like $\{(p,q), (p,r), (s,r), (s,t), \ldots, (v,w), (v,q)\}$. Here we note that if the $(k-1)^{th}$ and k^{th} cell are in the same row, then k^{th} and $(k+1)^{th}$ cell must be in the same column, but in different rows.

We now state following results without proofs

Result 5.4.1 *Every loop must have even number of cells.*

Result 5.4.2 *Any set of columns of A are linearly dependent if and only if the corresponding cells in the symbolic matrix contain a loop.*

Let us again consider the situation when we are trying to express a nonbasic column p_{ij} in terms of $(m+n-1)$ basic columns $\{p_{\alpha\beta}{}^{(B)}\}$. In view of Lemma (5.14), we have

$$p_{ij} = \sum_{(\alpha,\beta)} (\pm) p_{\alpha\beta}{}^{(B)},$$

which because of the fact that $p_{ij} = e_i + e_{m+j}$ tells that the number of basic cells on the RHS must be odd with '+' and '-' signs alternately, i.e.

$$p_{ij} = p_{iu}{}^{(B)} - p_{vu}{}^{(B)} + p_{vt}{}^{(B)} - \ldots - p_{ws}{}^{(B)} + p_{wj}{}^{(B)}. \tag{5.18}$$

Since existence of loop means linear dependence, and every nonbasic column can be written uniquely in terms of basic columns, there will exist one and only loop for the nonbasic cell (i,j). In a certain sense it is the result which says that every vector of a finite dimensional vector space can be written uniquely in terms of the basis vectors.

In (5.18), as a matter of convention, it is understood that those cells which are not in the loop, for them y_{ij} value is zero.

5.5 Optimality Conditions/ Stopping Criterion

In the usual simplex algorithm we stop when all $z_j - c_j \geq 0$ (for the maximization case). So the next objective is to find out the analogue of $(z_j - c_j)$ for the transportation case. For this we write the dual of the given balanced transportation problem (5.6). This dual is

$$\text{Max} \quad \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j$$

subject to

$$\begin{aligned} u_i + v_j &\leq c_{ij} \quad (i = 1,2,\ldots,m;\ j = 1,2,\ldots,n) \\ u_i \text{ and } v_j &\text{ unrestricted in sign.} \end{aligned} \tag{5.19}$$

Let $X = \{x_{ij}\}$ be a feasible solution of the primal problem (5.6) and $\{(u,v),\ i = 1,2,\ldots,m;\ j = 1,2,\ldots,n\}$ be a feasible solution of the dual problem (5.19), then for optimality

$$\sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}\ ,$$

i.e.

$$\sum_{i=1}^{m} u_i\left(\sum_{j=1}^{n} x_{ij}\right) + \sum_{j=1}^{n} v_j\left(\sum_{i=1}^{m} x_{ij}\right) - \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} = 0\ ,$$

i.e.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(u_i + v_j - c_{ij})x_{ij} = 0. \tag{5.20}$$

But $x_{ij} \geq 0$ and $u_i + v_j \leq c_{ij}$, hence (5.20) gives

$$(u_i + v_j - c_{ij})x_{ij} = 0 \quad \text{for all i and j.}$$

Therefore for $x_{ij} > 0$ we have $(u_i + v_j - c_{ij}) = 0$, otherwise $(u_i + v_j - c_{ij}) \leq 0$ as u_i, v_j are certainly dual feasible. But $x_{ij} > 0$ means that $(i,j)^{th}$ cell is a basic cell. Hence to check the optimality of the current basic feasible solution, we first determine u_i and v_j $(i = 1,\ldots,m;\ j = 1,\ldots,n)$ such that for each basic cell, $u_i + v_j = c_{ij}$. Then we evaluate $u_i + v_j - c_{ij}$ for all nonbasic cells. If $(u_i + v_j - c_{ij}) \leq 0$ for each nonbasic cell, then the current basic feasible solution is optimal for the given balanced transportation problem.

Determination of u_i $(i = 1,2,\ldots,m)$ ***and*** v_j $(j = 1,2,\ldots,n)$

We must now address the question of finding u_i and v_j such that $u_i + v_j = c_{ij}$ for each basic cell. As there are only $(m + n - 1)$ basic cells and we have $m + n$ unknowns $(u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n)$ the system of equations

$$u_i + v_j = c_{ij}, \tag{5.21}$$

for all i and j such that the cell (i,j) is a basic cell, can not have unique solution. Infact out of $(m + n)$ unknowns, one unknown has to be chosen arbitrary. In practice we take $u_1 = 0$ and then use the given set of equations (5.21) to determine other $(m + n - 1)$ unknowns.

5.6 Methods for Finding a Starting Solution

Once we know to compute $y_{ij}^{(\alpha,\beta)}$ and the stopping criterion, we can implement the transportation algorithm provided we have a suitable method for finding the starting basic feasible solution. Unlike the simplex method for LPP, because of the special structure of the balanced transportation problem, there are very simple and efficient methods for finding the starting solution. Some of these methods are

(i) North -West Corner Rule
(ii) Method of Column Minima
(iii) Method of Row Minima
(iv) Method of Matrix Minima
(v) VAM(Vogel's Approximation Method) .

In the following, we describe each of the above methods and illustrate through an example.

(i) **North -West Corner Rule**

Let the $(m \times n)$ transportation tableau be given as,

	b_1	b_2	.....	b_n
a_1	c_{11}	c_{12}	.....	c_{1n}
a_2	c_{21}	c_{22}	.....	c_{2n}
⋮	⋮	⋮		⋮
a_m	c_{m1}	c_{m2}	.....	c_{mn}

We start from the north-west corner of the tableau. We then compare a_1 and b_1, i.e. find the $\min(a_1, b_1)$. If this minimum is a_1 then we take $x_{11} = a_1$ and cross the first row because there is nothing is left at the first source. However if $\min(a_1, b_1) = b_1$ then we take $x_{11} = b_1$ and cross the first column because the requirement of the first destination is fulfilled.

We next consider the north-west corner of the resulting tableau after the first source has been deleted and b_1 is updated to $b_1 - x_{11}$ (this will happen when $\min(a_1, b_1) = a_1$) or the first destination has been deleted and a_1 is updated to $a_1 - x_{11}$ (this will happen when $\min(a_1, b_1) = b_1$, and then repeat the procedure as described earlier. As the problem is balanced $\left(\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j\right)$, in the end (the north-west corner rule of the final resulting transportation tableau) the availabilities and the requirements will match exactly and that common value will be taken as x_{ij} for that cell.

Once we determine values x_{ij} in this manner, we can prove that the corresponding cells (i, j) in the symbolic matrix will constitute a set of basic cells or equivalently the corresponding columns p_{ij} in A will be linearly independent. Further, in the absence of degeneracy, there will be exactly $(m + n - 1)$ such basic cells and so the corresponding $(m + n - 1)$ columns of A will give the basis matrix for which the solution so obtained is basic feasible.

Example 5.6.1 *Use the north west corner rule to find a starting solution to the following balanced transportation problem*

	13	18	14
10	2 (10)	1	4
15	6	3	2
20	4	2	3

Solution The given problem is balanced because $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = 45$. Now we start the N-W corner rule and compute $\min(a_1, b_1) = \min(10,13) = 10$, so we take $x_{11} = 10$, delete the first source and update the $b_1 = b_1 - x_{11} = 13 - 10 = 3$. This gives the resulting tableau as

	3	18	14
15	6 (3)	3	2
20	4	2	3

As the $\min(15,3) = 3$, we take $x_{21} = 3$, delete the first destination and update a_2 to $a_2 - x_{21} = 15 - 3 = 12$. This gives the resulting tableau as

	18	14
12	3 (12)	2
20	2	3

Again $\min(12, 18) = 12$ so we take $x_{22} = 12$, delete the second source and update $b_2 = 18 - 12 = 6$. This gives the resulting tableau as

	6	14
20	2 (6)	3

Here $\min(20, 6) = 6$, so we take $x_{32} = 6$, delete the second destination and update $a_3 = a_3 - x_{32} = 20 - 6 = 14$. This gives the final resulting tableau as

		14
14		3 (14)

so we take $x_{33} = 14$.

In practice we need not do the above steps for each resulting matrix separately. Infact all these steps can be done in the first $(m \times n)$ transportation tableau itself, to get the following

	13	18	14
10	2 (10)	1	4
15	6 (3)	3 (12)	2
20	4	2 (6)	3 (14)

Therefore the solution ($x_{11} = 10$, $x_{12} = 3$, $x_{22} = 12$, $x_{32} = 6$, $x_{33} = 14$, other $x_{ij} = 0$) is certainly a feasible solution for the given balanced transportation problem. Infact it is also a basic feasible solution because the corresponding cells in the symbolic matrix, namely (1,1), (2,1), (2,2), (3,2) and (3,3) constitute a set of basic cells as shown below

p_{11}	p_{12}	p_{13}
p_{21}	p_{22}	p_{23}
p_{31}	p_{32}	p_{33}

Further, in the absence of degeneracy these basic cells will precisely be $(m+n-1)$ in number, thereby giving $(m+n-1)$ linearly independent columns of A. In this example the columns p_{11}, p_{21}, p_{22}, p_{32}, and p_{33} are five linearly independent columns of A.

Though we are not giving a formal proof, it can be verified that these columns of A are linearly independent because there exists no loop amongst any subset of five basic cells as identified in the symbolic matrix.

(ii) **Method of Column Minima**

In the north west corner rule, the starting basic feasible solution is obtained taking into consideration the numbers a_i and b_j only and the cost elements c_{ij} are not used at all. As our aim is to get a basic feasible solution for which the total transportation cost is minimum, it is expected that if the cost elements are also taken into consideration (along with a_i and b_j) then probably the starting basic feasible solution will be nearer to the optimal basic feasible solution and therefore the solution can be obtained in lesser number of iterations. Here we wish to emphasize that this is just an expectation and there is no guarantee for the same. The methods of column minima, row minima, matrix minima and VAM, are all in this category, i.e for finding the starting basic feasible solution they take into consideration a_i, b_j and also c_{ij}.

We now describe the column minima method, the description of others will then be analogous.

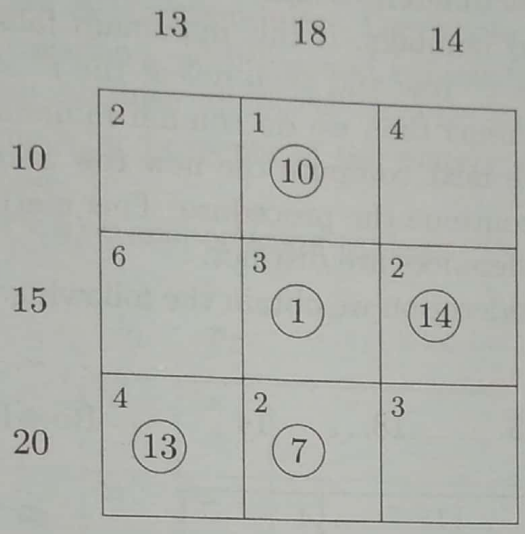
We choose the first column and in that column we choose the smallest cost element c_{i1}. Then we compute $\min(a_i, b_1)$. If this minimum is b_1 then we take $x_{i1} = b_1$ and delete the first destination and proceed to the second column to repeat the procedure. However, if this minimum is a_i, then we take $x_{i1} = a_i$, update b_1 to $b_1 - x_{i1}$ and then choose the next smallest element in the first column to repeat the procedure. Once the requirement of the first destination is fulfilled, we proceed to the second column and continue the procedure.

For the balanced transportation problem given in Example (5.6.1), the starting basic feasible solution given by the column minima method is

	13	18	14
10	2 (10)	1	4
15	6	3 (1)	2 (14)
20	4 (3)	2 (17)	3

(iii) **Method of Row Minima**

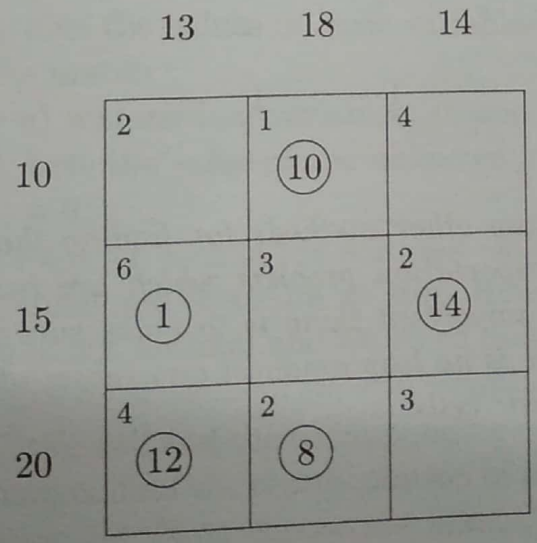
This is exactly same as the method of column minima, except that we change the word 'column' by row. For Example (5.6.1), the method of row minima gives the following starting basic feasible solution



(iv) **Method of Matrix Minima**

This method is again very similar to the earlier two methods, namely 'column minima' and 'row minima'. Here we choose the smallest cost element (say c_{ij}) in the whole matrix C and find $\min(a_i, b_j)$. If $\min(a_i, b_j) = a_i$ then we take $x_{ij} = a_i$, drop the ith source and update b_j to $b_j - x_{ij}$. Similarly, if $\min(a_i, b_j) = b_j$, we take $x_{ij} = b_j$, drop the jth source and update a_i to $a_i - x_{ij}$. We next find the smallest element in the resulting cost matrix and continue.

For the example under consideration, the method of matrix minima gives the following starting basic feasible solution



(v) **VAM(Vogel's Approximation Method)**

Here we first find the row differences and column differences for each row and column of the cost matrix C. For a given row(column) the row difference(column difference) is defined as the difference of the smallest and the next smallest numbers in that row(column). Once these m row differences and n column differences are known, we find the maximum of these $(m + n)$ numbers. If this maximum falls for a row(say i^{th} row) then we do row minima in the ith row and then delete the ith source. If this minimum falls for a column(say j^{th} column) then we do column minima in the jth column and then delete the jth column. We next compute the new row and column differences for the reduced cost matrix and continue the procedure. This method is best suited if the row differences and column differences are distinct.

For the example under consideration we obtain the following starting bfs using VAM

	13	18	14	Row Differences	
10	2 (10)	1	4	1	
15	6	3 (1)	2 (14)	1	1
20	4 (3)	2 (17)	3	1	1
Column differences	2	1	1		
		1	1		

Remark 5.6.1 *There are many other methods for finding the starting basic feasible solution for the balanced transportation problem which are based on logic similar to those of described above. The important thing is to make sure that we get cells in the symbolic matrix such that there is no loop amongst any subset of these cells so that they constitute the desired set of basic cells.*

5.7 The Complete Algorithm

We now have the complete machinery to describe the transportation algorithm for solving the balanced transportation problems (5.6). As explained earlier, this algorithm is essentially the simplex algorithm whose implementation gets simplified due to the fact that the coefficient matrix A is unimodular. The whole algorithm is essentially based on the operations of comparison, addition and subtraction; and there is no operation of multiplication/division because all $y_{ij}^{(\alpha,\beta)} = +1, -1$ or 0, a property which is a consequence of the unimodular property of the matrix A. The stepwise description of the algorithm is as follows.

Step 1 Consider the $(m \times n)$ transportation tableau

	b_1	b_2	.....	b_n
a_1	c_{11}	c_{12}	.....	c_{1n}
a_2	c_{21}	c_{22}	.....	c_{2n}
⋮	⋮	⋮		⋮
a_m	c_{m1}	c_{m2}	.....	c_{mn}

and obtain the starting solution by employing any of the methods described in Section 5.6 (Step 1 will not only give the values of basic variables but also identify $(m + n - 1)$ basic cells in the symbolic matrix).

Step 2 Determine $(m + n)$ scalars(dual variables) u_i and v_j such that $u_i + v_j = c_{ij}$ for basic cells. As explained, here the value of one unknown is to be chosen arbitrarily and in practice, we choose $u_1 = 0$.

Step 3 Evaluate $(u_i + v_j - c_{ij})$ for each of the nonbasic cells. If all $(u_i + v_j - c_{ij}) \leq 0$, then the current solution is optimal and therefore we stop. However, if some $(u_i + v_j - c_{ij}) > 0$ then the current solution is not optimal and we go to Step 4.

Step 4 Choose the positive most value of $(u_i + v_j - c_{ij})$. Let this be $(u_r + v_s - c_{rs})$. The $(r,s)^{th}$ cell will become a basic cell and the corresponding column p_{rs} will enter the basis.

Step 5 Using symbolic matrix, find the representation of $(r,s)^{th}$ cell in terms of the basic cells as explained in Section 5.4. Next choose the $\min(x_{ij}^{(B)})$ over those (i, j) for which $y_{ij}^{(\alpha,\beta)} = +1$, i.e. find the minimum of values of those basic variables $(x_{ij}^{(B)})$ for which

'+' sign is attached in the loop. Let this correspond to the cell (u, v). Then the cell (u, v) will become a nonbasic cell and the corresponding column p_{uv} will leave the basis.

Step 6 Find the new b.f.s and go to Step 2. The new solution is obtained by using the formula

$$\hat{x}_{\alpha\beta}^{(B)} = x_{\alpha\beta}{}^{(B)} - y_{rs}{}^{(\alpha\beta)} x_{uv}{}^{(B)}, \quad (\alpha, \beta) \neq (u, v),$$

$$\hat{x}_{rs}^{(B)} = x_{uv}{}^{(B)},$$

i.e. in the new tableau, the value $x_{uv}{}^{(B)}$ is simply shifted at the $(r, s)^{th}$ cell, it is added to all those cells in the loop for which (-) sign is attached ($y_{rs}{}^{(\alpha\beta)} = -1$) and subtracted to all those cells in the loop for which (+) sign is attached ($y_{rs}{}^{(\alpha\beta)} = +1$). Those cells (i, j) which are not in the loop, the value $x_{ij}{}^{(B)}$ remains unchanged as for them $y_{rs}{}^{(\alpha\beta)} = 0$.

Remark 5.7.1 *As the transportation algorithm is essentially the simplex algorithm, we will obtain optimal solution in finite number of iterations. Here we may note that if $c_{ij} \geq 0$ then the transportation problem cannot have unbounded solution because it is a minimization problem and $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \geq 0$ for x_{ij} feasible.*

Remark 5.7.2 *We make a very important observation about the transportation problem. If a_i $(i = 1, 2, \ldots, m)$ and b_j $(j = 1, 2, \ldots, n)$ are integers then the starting solution is also in terms of integers only because there we are essentially comparing two integers to find $min(a_i, b_j)$. In the subsequent steps, as all y_{ij} are +1, -1 or 0, there is no division/multiplication in the algorithm and each time either two integers are being added or subtracted. Therefore the initial b.f.s as well as all subsequent b.f.s are in terms of integers only. This shows that the optimal solution is also in terms of integers only. The main reason for this property to hold is that the coefficient matrix A of a balanced transportation problem is unimodular.*

We now illustrate the working of the transportation algorithm with the help of some examples.

Example 5.7.1 *Solve the following transportation problem by finding the starting solution using the north west corner rule.*

	13	18	14
10	2	1	4
15	6	3	2
20	4	2	3

Solution The first iteration of the algorithm is

Step 1 The north west corner rule gives the following starting b.f.s.

	13	18	14
10	2 (10)	1	4
15	6 (3)	3 (12)	2
20	4	2 (6)	3 (14)

Step 2 Find u_1, u_2, u_3, v_1, v_2, v_3 such that $u_i + v_j = c_{ij}$ for each basic cell, i.e $u_1 + v_1 = 2$, $u_2 + v_1 = 6$, $u_2 + v_2 = 3$, $u_3 + v_2 = 2$ and $u_3 + v_3 = 3$. Taking $u_1 = 0$, we get $v_1 = 2$, $u_2 = 6 - 2 = 4$, $v_2 = 3 - u_2 = 3 - 4 = -1$, $u_3 = 2 - v_2 = 2 - (-1) = 3$ and $v_3 = 3 - u_3 = 3 - 3 = 0$.

	13	18	14	
10	2 (10)	1	4	u_1=0
15	6 (3)	3 (12)	2	u_2=4
20	4	2 (6)	3 (14)	u_3=3
	v_1=2	v_2= -1	v_3=0	

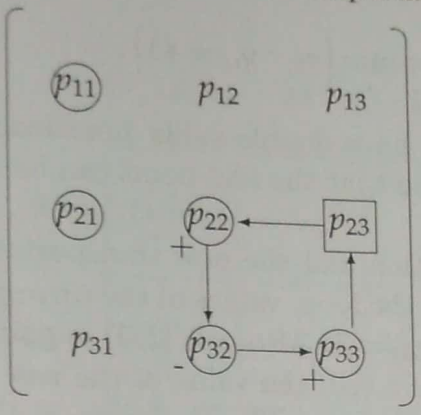
Step 3 We next compute $u_i + v_j - c_{ij}$ for each nonbasic cell. These are $u_1 + v_2 - c_{12} = 0 - 1 - 1 = -2$, $u_1 + v_3 - c_{13} = 0 + 0 - 4 = -4$, $u_2 + v_3 - c_{23} = 4 + 0 - 2 = 2$ and $u_3 + v_1 - c_{31} = 3 + 2 - 4 = 1$. We write these numbers in the right bottom most corner of each cell.

	13	18	14
10	2 (10)	1 -2	4 -4
15	6 (3)	3 (12)	2 2
20	4 1	2 (6)	3 (14)

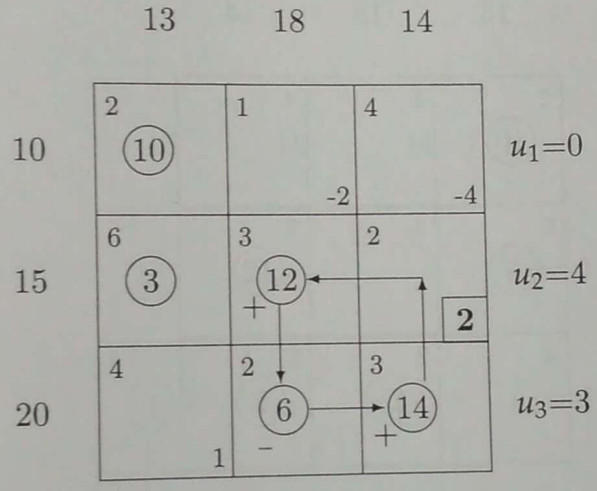
As not all $u_i + v_j - c_{ij}$ are ≤ 0, the current solution is not optimal.

Step 4 The positive most value of $u_i + v_j - c_{ij}$ is $u_2 + v_3 - c_{23} = 2$. So column p_{23} enters the basis and the cell $(2,3)$ becomes a basic cell.

Step 5 To find the column to leave the basis(or equivalently the basic cell which becomes nonbasic), we find the expression of the nonbasic cell $(2,3)$ in terms of basic cells as shown below



If there is no confusion then we make the loop in the tableau itself but we must remember that loop is in the symbolic matrix only. Therefore making the loop in the tableau itself we get



Now we identify those cells in the loop for which '+' sign is attached(i.e y_{ij} value is +1) and choose the one for which x_{ij} is least. Here it is $x_{22} = 12$ so cell $(2,2)$ becomes nonbasic and the corresponding column p_{22} leaves the basis.

Here we must note that it is essentially the usual minimum ratio criterion which is being used to identify the column which is going to leave the basis. We recall that the usual criterion is

$$\frac{x_{B_r}}{y_{rj}} = \min_i \left\{ \frac{x_{B_i}}{y_{ij}} : y_{ij} > 0 \right\},$$

which because $y_{rj} > 0$ means $y_{rj} = +1$. Therefore for the transportation problem the minimum ratio criterion becomes

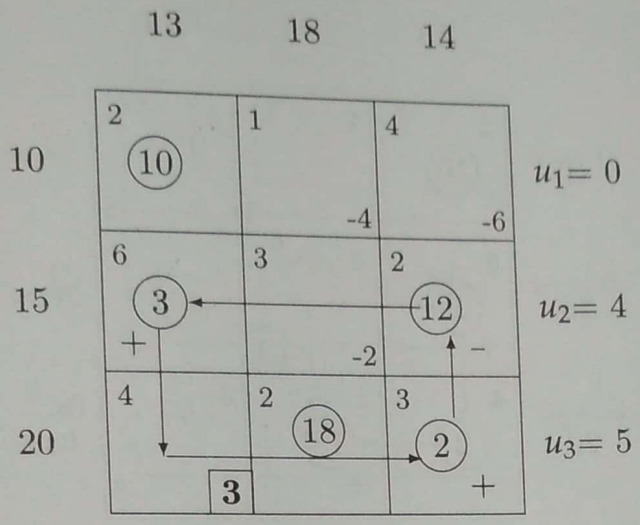
$$x_{B_r} = \min_i \left\{ x_{B_i} : y_{ij} = +1 \right\}.$$

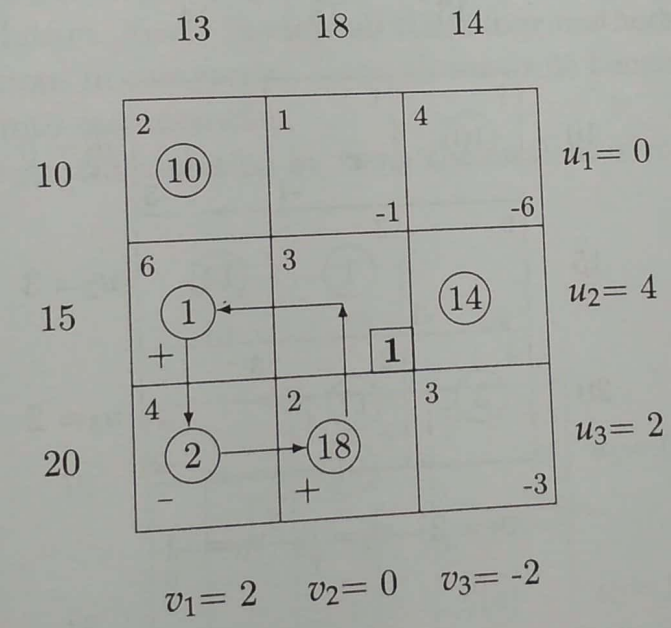
In the above, of course we should have double suffix notations of the transportation problem but we have avoided that so that the real point can be understood much more easily.

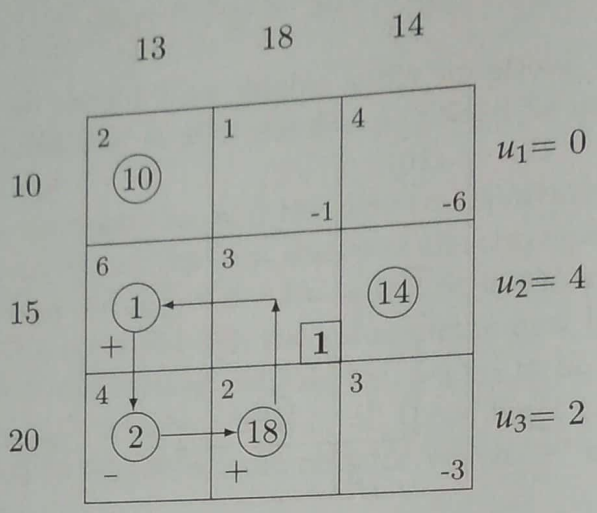
Step 6 We now find the new solution and the new transportation tableau. Since the basic cells $(1,1)$ and $(2,1)$ are not in the loop, values of the corresponding basic variables will not change, i.e $x_{11} = 10$ and $x_{21} = 3$. Also cell $(2,3)$ is going to be the new basic cell and $(2,2)$ is going to be nonbasic cell, the value of the new basic variable x_{23} will be the same as the value of the leaving variable x_{22}, i.e $x_{23} = 12$. For other basic cells in the loop, the value $x_{22} = 12$ will be added to all those basic cells for which '-' sign is attached and subtracted to all those basic cells for which '+' sign is attached. This gives the following tableau

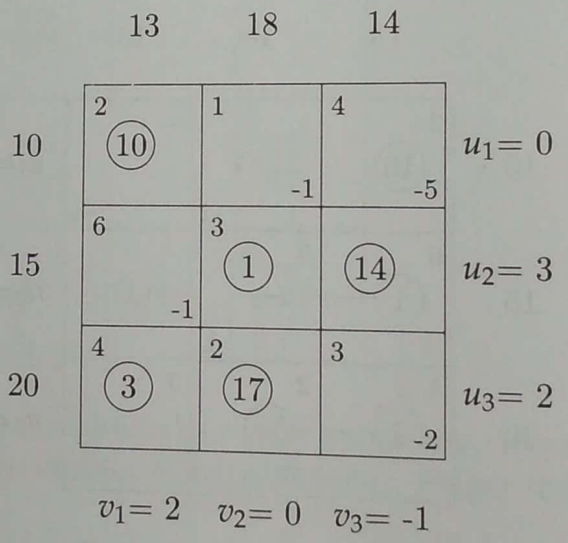
	13	18	14
10	2 (10)	1	4
15	6 (3)	3	2 (12)
20	4	2 (18)	3 (2)

We now go to next iteration and repeat the procedure. In practice we preform all the above steps in the same tableau. For our example, we get the following tableaus at the subsequent iterations







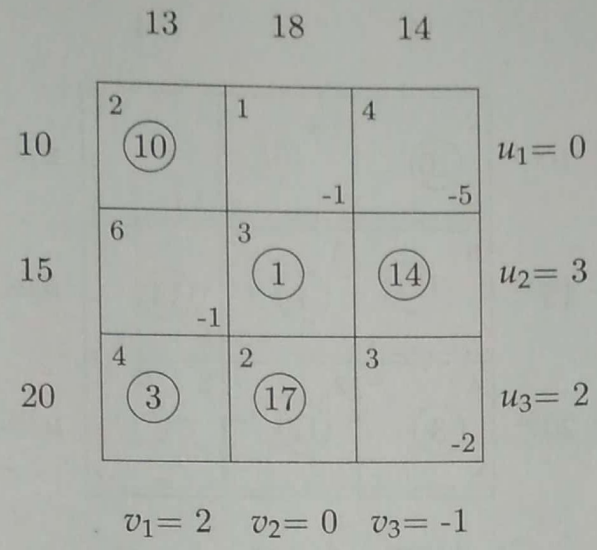


Now in the last tableau all $(u_i + v_j - c_{ij})$ are non positive and hence we stop. The current solution namely $x_{11}^* = 10$, $x_{22}^* = 1$, $x_{23}^* = 14$, $x_{31}^* = 3$, $x_{32}^* = 17$ is an optimal solution and $z^* = (2 \times 10) + (3 \times 1) + (2 \times 14) + (4 \times 3) + (2 \times 17) = 97$ is the optimal value.

Example 5.7.2 *For the problem of Example 5.7.1, obtain the optimal solution by finding the starting solution using (i) column minima (ii) row minima (iii) matrix minima (iv) VAM.*

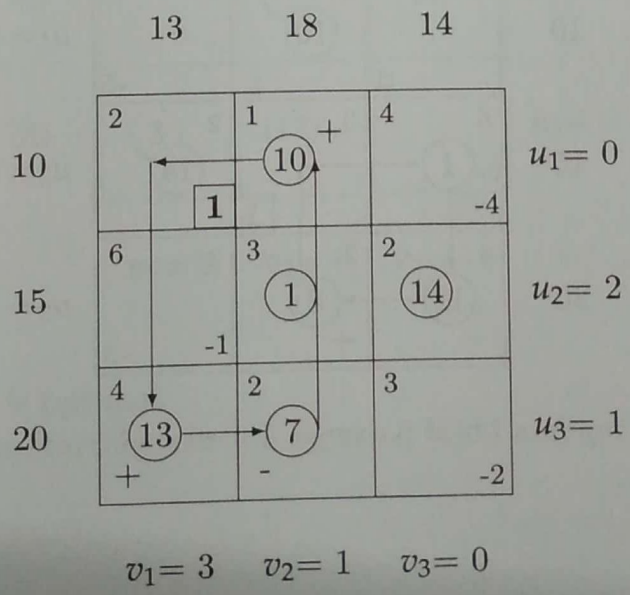
Solution

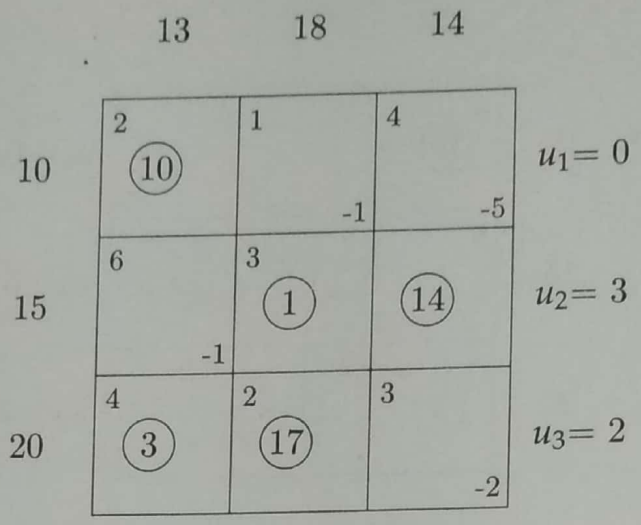
(i) Here we find the starting solution by using the method of column minima and get the following



As all $(u_i + v_j - c_{ij}) \leq 0$, the starting solution itself is optimal. From here we should not get the impression that this is always going to happen if column minima method is used to find the starting solution. Never the less all these four methods will, in general, take lesser number of iterations in comparison to north west rule because these methods take cost elements c_{ij} also into consideration.

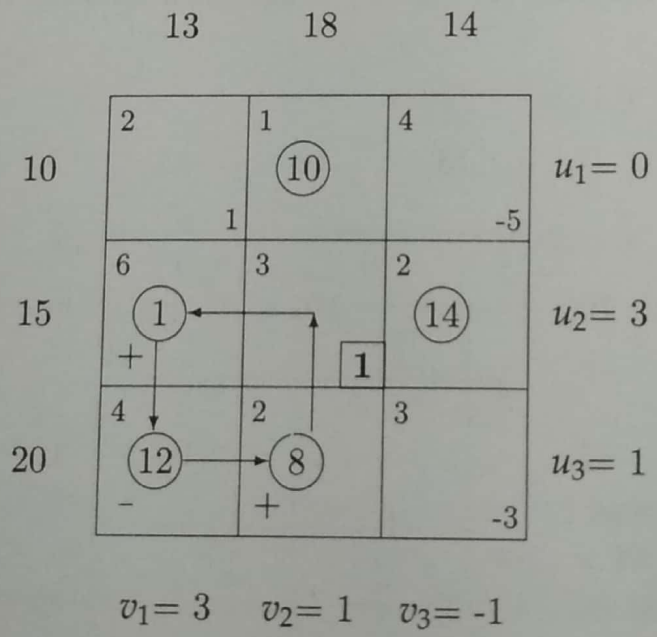
(ii) Here we find the starting solution by using the method of row minima and get the following tableaus.

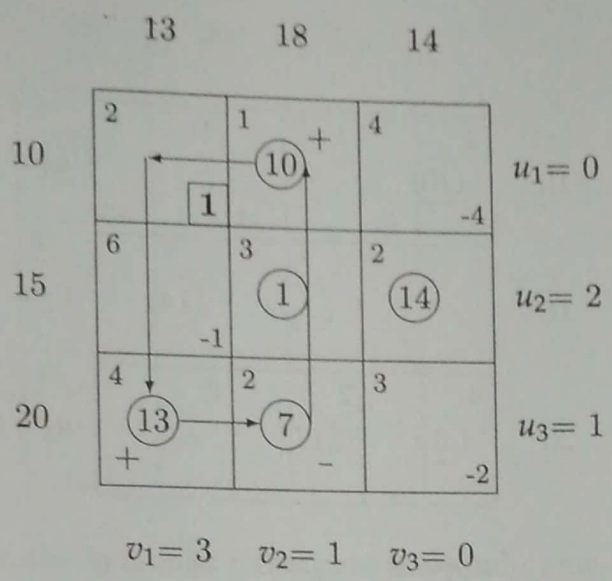


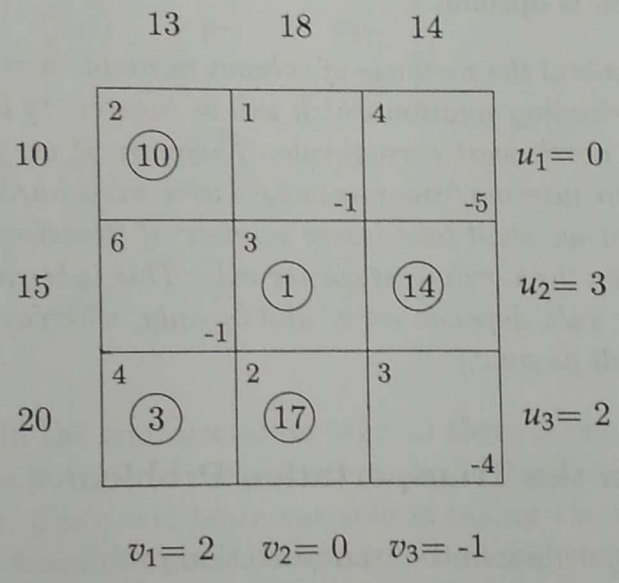


As all $(u_i + v_j - c_{ij}) \leq 0$, the current solution is optimal.

(iii) Here we find the starting solution by using the method of matrix minima and get the following tableaus

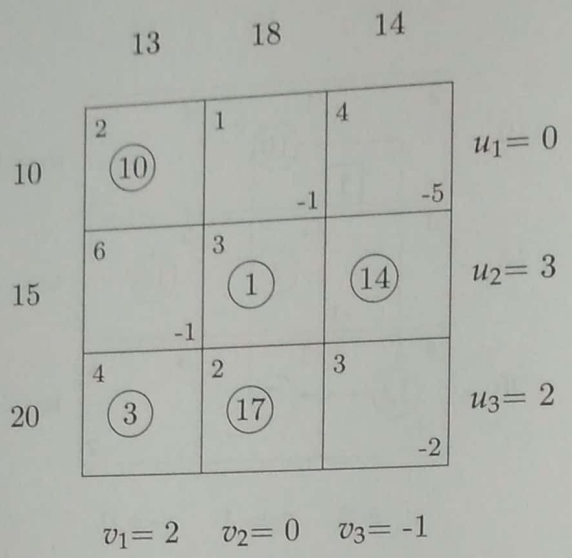






So the current solution is optimal.

(iv) Here we obtain the starting solution by using VAM and get the following tableaus.



So the current solution is optimal.

Remark 5.7.3 *In general the methods of column minima, row minima, matrix minima and VAM will give a starting solution which will be 'nearer' to the optimal solution than the one given by the north west corner rule. Therefore if we find starting solution by the methods of column minima/row minima/matrix minima/VAM and then solve the problem we expect that we shall take lesser number of iterations than the one when we find starting solution by the north west corner rule. This is because, as explained earlier, the north west corner rule depends on a_i and b_j only, whereas the other four methods depend on a_i, b_j as well as on c_{ij}.*

5.8 Degeneracy in the Transportation Problem

To understand the degeneracy in the transportation problem, let us consider the following balanced transportation problem and find its starting solution by using the north west corner rule to get the tableau

	10	20	15
10	2 (10)	1	4
15	6	3 (15)	2
20	4	2 (5)	3 (15)

Here we note that in the symbolic matrix we have only four basic cells where as we need $(m+n-1) = 3+3-1 = 5$ basic cells, i.e.

$$\begin{pmatrix} (p_{11}) & p_{12} & p_{13} \\ p_{21} & (p_{22}) & p_{23} \\ p_{31} & (p_{32}) & (p_{33}) \end{pmatrix}$$

This indicates that in the transportation tableau there is one more basic variable at the zero level which has got mixed with other nonbasic variables which are any way at zero level by definition. Since one basic variable is taking the zero value, the current b.f.s is degenerate. But before we start performing the usual steps of the transportation algorithm we need to identify that basic cell for which the value of the basic variable is zero. In other words, we have to identify an additional basic cell in the symbolic matrix and take the corresponding value of the basic variable as zero. For this we use the property that in any transportation tableau(or equivalently in the corresponding symbolic matrix) the set of $(m+n-1)$ basic cells is *connected*, i.e. given any two basic cells (s,t) and (u,v) there is always a path $\{(s,t), (s,r), (p,r), \ldots, (u,v)\}$ between them. Geometrically this means that we can start from the cell (s,t) and moving along row/column via basic cells we reach the cell (u,v).

In case the given b.f.s is degenerate, we have less than $(m+n-1)$ basic cells. Therefore there will certainly be some part which is disconnected. In our example, there is no path

between cells $(1,1)$ and $(2,2)$. So we have to add a cell at zero level in the tableau so that by taking that additional cell, the set of $(m+n-1)$ cells becomes connected. It is not difficult to see that we can take this cell to be $(1,2)$ or $(2,1)$; infact there are many other possibilities as well. If we decide for the cell $(1,2)$ then we take $x_{12}=0$ as a basic variable and make the cell $(1,2)$ as a basic cell, giving the following b.f.s of $(m+n-1)$ basic cells and start the transportation algorithm as usual.

	10	20	15
10	2 (10)	1 (0)	4
15	6	3 (15)	2
20	4	2 (5)	3 (15)

The main reason for getting one basic cell short is that as $\min(a_1, b_1) = \min(10,10) = 10$, we meet the source constraint as well as the destination constraint simultaneously, thereby deleting the first source and first destination at the same time. Whenever this happens we shall always be short of one basic cell. Therefore depending upon the situation, we may have one or more number of basic cells less than the required number $(m+n-1)$ and we will have to introduce appropriate number of basic cells at zero level as explained above to get the $m+n-1$ basic variables so that the algorithm could be started.

While adding a cell at zero level and encircling it so that it gets differentiated from other nonbasic cells at zero level is the correct way, it may create some problem in the implementation. If we are solving the problem by hand then we can always differentiate between two 'zeros', one which is encircled and the other which is not(for a nonbasic variable), but there may be some problem if we are working on a machine. Therefore in practice we take the value of the degenerate basic variable as a small number ϵ(say 0.01) and make the appropriate modifications as shown below and implement the algorithm,

	10	$20+\epsilon$	15
$10+\epsilon$	2 (10)	1 (ϵ)	4
15	6	3 (15)	2
20	4	2 (5)	3 (15)

5.9 The Unbalanced Transportation Problem

We have defined a transportation problem as a balanced transportation problem if $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. A transportation problem which is not balanced is termed as an *unbalanced transportation problem.* Therefore the general transportation problem

$$\text{Min} \qquad z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le a_i \qquad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j \qquad (j = 1,2,\ldots,n)$$

$$x_{ij} \ge 0 \qquad (i = 1,2,\ldots,m; j = 1,2,\ldots,n)\,, \qquad (5.22)$$

will be unbalanced if $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$ or $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$. We shall discuss each of these two cases separately. In our discussion, we shall be assuming that $c_{ij} \ge 0$ for all i and j.

Result 5.9.1 *Let $c_{ij} \ge 0$ for all i and j. Let (x_{ij}^*) be an optimal solution of (5.22) then*

$$\sum_{i=1}^{m} x_{ij}^* = b_j, \quad (j = 1,2,\ldots,n).$$

Though a formal mathematical proof can be given, the result is obvious because transporting any quantity more than b_j $(j = 1, 2, \ldots, n)$ to the jth destination will incurr additional cost(as $c_{ij} \geq 0$) and therefore it cannot give optimal solution. Since requirement of the jth destination is b_j or more and we wish to minimize the cost of transportation we should not give any thing more than b_j as $c_{ij} \geq 0$ for all i and j.

Case 1 Let $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$

In the following we shall show that in this case, the given unbalanced transportation problem (5.22) can be transformed into a new transportation problem which is balanced and therefore can be solved by the usual transportation algorithm discussed in Section 5.7. It is obvious that for Case 1 problem (5.22) is certainly feasible and has an optimal solution. In view of Result 5.9.1, problem (5.22) is equivalent to

$$\text{Min} \qquad z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \leq a_i \quad (i = 1, 2, \ldots, m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1, 2, \ldots, n)$$

$$x_{ij} \geq 0 \quad \text{for all } i \text{ and } j\,. \tag{5.23}$$

We now introduce m slack variables $x_{i,n+1}$ $(i = 1, 2, \ldots, m)$ in each of the source constraints of (5.23) to get

$$\text{Min} \qquad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + 0x_{1,n+1} + \ldots + 0x_{m,n+1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} + x_{i,n+1} = a_i \quad (i = 1, 2, \ldots, m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1, 2, \ldots, n)$$

$$x_{ij} \geq 0 \quad \text{for all } i \text{ and } j\,, \tag{5.24}$$

which can be written as

$$\text{Min} \quad \sum_{i=1}^{m}\sum_{j=1}^{n+1} c_{ij}\, x_{ij} \quad (c_{i,n+1} = 0,\ i = 1,2,\ldots,m)$$

subject to

$$\sum_{j=1}^{n+1} x_{ij} = a_i \quad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1,2,\ldots,n)$$

$$x_{ij} \geq 0 \quad \text{for all } i \text{ and } j\,. \tag{5.25}$$

But $x_{i,n+1} = a_i - \sum_{j=1}^{n} x_{ij}$ and hence

$$\begin{aligned}\sum_{i=1}^{m} x_{i,n+1} &= \sum_{i=1}^{m} a_i - \sum_{i=1}^{m}\left(\sum_{j=1}^{n} x_{ij}\right)\\ &= \sum_{i=1}^{m} a_i - \sum_{j=1}^{n}\left(\sum_{i=1}^{m} x_{ij}\right)\\ &= \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j\\ &= b_{n+1}(\geq 0) \quad \text{(say)}.\end{aligned} \tag{5.26}$$

In view of equation (5.26), problem (5.25) can be rewritten as

$$\text{Min} \quad z = \sum_{i=1}^{m}\sum_{j=1}^{n+1} c_{ij}\, x_{ij} \quad (c_{i,n+1} = 0,\ i = 1,2,\ldots,m)$$

subject to

$$\sum_{j=1}^{n+1} x_{ij} = a_i \quad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1,2,\ldots,n,n+1)$$

$$x_{ij} \geq 0 \quad (i = 1,2,\ldots,m;\ j = 1,2,\ldots,n,n+1)\,, \tag{5.27}$$

which is a balanced transportation problem.

Looking at the above problem, we note that if $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$ and all $c_{ij} \geq 0$, then to make the problem balanced, we introduce a dummy destination $(n+1)$ whose

requirement is $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ and all cost elements $c_{i,n+1} = 0$ for $i = 1, 2, \ldots, m$. In practical terms, this amounts to introduce a dummy column in the tableau having all cost elements as zero and requirement as the excess commodity available, which is $\sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$. This balanced transportation problem (5.27) can then be solved by the usual transportation algorithm and the optimal solution of the unbalanced problem (5.23) could be obtained by ignoring the components $x^*_{i,n+1}$ which appear in the $(n+1)^{th}$ column.

Example 5.9.1 *Solve the following transportation problem*

	12	15	14
10	2	1	4
15	6	3	2
20	4	2	3

Solution As $\sum_{i=1}^{m} a_i = 45$ and $\sum_{j=1}^{n} b_j = 41$, the given transportation problem is unbalanced. But as all $c_{ij} \geq 0$ and $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, we can make this transportation problem balanced and then solve it. As discussed here, to make this transportation problem balanced we have to introduce a 4^{th}(dummy) destination(column) with $b_4 = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j = 45 - 41 = 4$ and $c_{14} = c_{24} = c_{34} = 0$. This gives the following balanced transportation problem which could be solved easily

	12	15	14	4
10	2	1	4	0
15	6	3	2	0
20	4	2	3	0

The optimal solution of the above balanced transportation problem is

	12	15	14	4
10	2 (10)	1	4	0
15	6	3	2 (14)	0 (1)
20	4 (2)	2 (15)	3	0 (3)

Therefore the optimal solution of the original(unbalanced transportation problem) is $x_{11}^* = 10$, $x_{23}^* = 14$, $x_{31}^* = 2$, $x_{32}^* = 15$ and other $x_{ij}^* = 0$. Further, source 2 will have excess of one unit and source 3 will have excess of 3 units of the commodity.

Case 2 Let $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$

From the source constraints and the destination constraints of the transportation problem (5.22) we note that for feasibility $\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$. Therefore when $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$, the problem (5.22) is definitely infeasible and therefore there is no question of finding its optimal solution.

But Case 2 is the most common in reality so what best we can do in this scenario. Noting that we can not meet the requirement of every destination(customer) we attempt to find the most satisfying solution.

Let us recall that, as all $c_{ij} \geq 0$, we are solving the equivalent problem

$$\text{Min} \qquad z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \leq a_i \quad (i = 1, 2, \ldots, m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad (j = 1, 2, \ldots, n)$$

$$x_{ij} \geq 0 \quad \text{for all } i \text{ and } j\,, \tag{5.28}$$

which under the assumption $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$ is infeasible. Therefore here 'solving' is to be understood in terms of finding the best compromise solution.

Knowing that the requirement of every customer can not be fully met, we give priority to every customer(destination) by specifying penalty costs for each destination. Larger

the penalty is, the more priority the customer has, and hence less shortage is expected. Therefore we introduce a dummy source $(m+1)$ with availability $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$ and $c_{m+1,j} = p_j$ $(j = 1, 2, \ldots, n)$, where p_j indicates the penalty for not fulfilling the requirement of jth destination. As explained above if $p_j > p_k$ then priority for j^{th} customer is more in comparison to the kth customer and therefore it is expected that the shortage for j^{th} customer will be less than the shortage for the kth customer. As the $(m+1)^{th}$ source is a dummy source, the a_{m+1} units of commodity is NOT physically available - it is available on the paper only. So a destination having the most penalty will get the least from the $(m+1)^{th}$ source, and therefore will have the least shortage. If no priority is prescribed then all destinations are treated equally preferable so all $p_i's$ are taken equal, in particular $p_1 = p_2 = \ldots = p_n = 0$.

Example 5.9.2 *Solve the following transportation problem and interpret your result*

	13	18	14
10	2	1	4
15	6	3	2
12	4	2	3

Solution Here $\sum_{i=1}^{3} a_i = 37$ and $\sum_{j=1}^{3} b_j = 45$. As $\sum_{i=1}^{3} a_i \neq \sum_{j=1}^{3} b_j$ it is an unbalanced transportation problem. Also as $\sum_{i=1}^{3} a_i < \sum_{j=1}^{3} b_j$, the problem is infeasible, i.e. we can not have any solution which will meet the requirement of every destination.

As no priority is prescribed in the question, we can assume that all destinations are equally preferable and so introduce a dummy source 4 with $a_4 = \sum_{j=1}^{3} b_j - \sum_{i=1}^{3} a_i = 45-37 = 8$ and $p_1 = p_2 = p_3 = p_4 = 0$, i.e. we solve the following balanced transportation problem

	13	18	14
10	2	1	4
15	6	3	2
12	4	2	3
8	0	0	0

An optimal solution of the above balanced transportation problem is

	13	18	14
10	2 (5)	1 (5)	4
15	6	3 (1)	2 (14)
12	4	2 (12)	3
8	0 (8)	0	0

The solution from the above tableau is $x_{11}^* = 5$, $x_{12}^* = 5$, $x_{22}^* = 1$, $x_{23}^* = 14$, $x_{33}^* = 12$ and other $x_{ij}^* = 0$ $(i = 1,2,3;\ j = 1,2,3)$. As $a_4 = 8$ units are not physically available and these are supplied to the first destination (because $x_{41}^* = 8$), we infer that the requirement of the first destination will fall short by 8 units of commodity.

Example 5.9.3 *Solve the transportation problem given in Example 5.9.2 but with the additional information that the requirement of the first destination has to be met exactly.*

Solution As the demand of the first destination has to be met exactly and the given transportation problem is unbalanced with $\sum_{i=1}^{3} a_i < \sum_{j=1}^{3} b_j$, we have to prescribe penalties p_1, p_2 and p_3 with p_1 being a large positive number. We may take $p_1 = 100$ with $p_2 = p_3 = 0$ as no priority is given for the other two destinations. Therefore we have to solve the following balanced transportation problem

	13	18	14
10	2	1	4
15	6	3	2
12	4	2	3
8	100	0	0

An optimal solution of the above balanced transportation problem is obtained as

	13	18	14
10	2 (10)	1	4
15	6	3 (1)	2 (14)
12	4 (3)	2 (9)	3
8	100	0 (8)	0

From the above tableau, the solution of the original problem is obtained as $x_{11}^* = 10$, $x_{22}^* = 1$, $x_{23}^* = 14$, $x_{31}^* = 3$, $x_{32}^* = 9$ and other $x_{ij}^* = 0$ $(i = 1,2,3;\ j = 1,2,3)$. Further as $a_4 = 8$ is not available physically(it is just available on the paper only to make the transportation problem balanced) and $x_{42}^* = 8$, we infer that the requirement of the second destination will fall short by 8 units.

5.10 The Assignment Problem: Description and Mathematical Model

Though the assignment problem may be given many meaningful descriptions, traditionally it is described in terms of assigning n persons to n jobs in an 'optimal' manner. Let n persons be denoted by $P_1, P_2, \ldots, P_n$. In a similar manner we denote the n jobs by $J_1, J_2, \ldots, J_n$. Also let c_{ij} $(i = 1,2,\ldots,m;\ j = 1,2,\ldots,n)$ denote the amount which the

i^{th} person charges to complete the jth job. Here we also assume that only one person is to be assigned to one job and vice-versa. In assignment problem, we wish to determine the assignment of persons to the jobs so that the total cost of completing all the jobs is minimum.

To get a mathematical model of the assignment problem we introduce variables x_{ij} $(i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n)$ as follows

$$x_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ person is assigned the } j^{th} \text{ job} \\ 0 & \text{otherwise.} \end{cases}$$

Let $X = (x_{ij})$ be the $(n \times n)$ matrix with entries as x_{ij}. As only one person is to be assigned only one job and vice versa, if there is 'one' in the ith row and the jth column of the matrix X then all other entries in that row and column must be zero. Therefore, the mathematical description of the physical constraint that 'only one person should be assigned to one job and vice versa' is

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \ldots, n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \ldots, n)$$

$$x_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j. \tag{5.29}$$

We next consider the quantity to be optimized and express the same in terms of parameters(i.e. c_{ij}) and decision variables(i.e. x_{ij}). Here the quantity to be optimized is the 'total cost of completing all the jobs' which can be expressed as

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{5.30}$$

because in (5.30) the contribution from the term $c_{ij}x_{ij}$ is c_{ij} or 'zero' depending upon whether the ith person is assigned the jth job or not, and then we are summing only such c_{ij}'s. Therefore the mathematical model of the assignment problem(AP) is

$$\text{Min} \qquad z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

subject to

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} x_{ij} = 1 && (j = 1, 2, \ldots, n) \\ &\textstyle\sum_{j=1}^{n} x_{ij} = 1 && (i = 1, 2, \ldots, n) \\ &x_{ij} = 0 \text{ or } 1 && \text{for all } i \text{ and } j\,. \end{aligned} \tag{5.31}$$

Remark 5.10.1 *We can also think of the assignment problem as a facility-location problem. Let there be n facilities(e.g. fire stations, schools, petrol pumps etc) which are to be located in n physical locations of a city. Let c_{ij} denotes the cost/return of assigning the ith facility to the jth location. Our aim here is to find the optimal location of the given n facilities to the given n locations.*

We now have the following definitions.

Definition 5.10.1 (Doubly Stochastic Matrix). *An $(n \times n)$ matrix $D = (d_{ij})$ is called a doubly stochastic matrix if its entries are non-negative and all its row sums and column sums are unity, i.e.*

$$\sum_{i=1}^{n} d_{ij} = 1 \ (j = 1, 2, \ldots, n)$$

$$\sum_{j=1}^{n} d_{ij} = 1 \ (i = 1, 2, \ldots, n)$$

$$d_{ij} \geq 0 \text{ for all } i \text{ and } j .$$

Definition 5.10.2 (Permutation Matrix). *An $(n \times n)$ matrix $P = (p_{ij})$ is called a permutation matrix if its entries are 'zero' or 'one' and all its row sums and column sums are unity, i.e.*

$$\sum_{i=1}^{n} p_{ij} = 1 \ (j = 1, 2, \ldots, n)$$

$$\sum_{j=1}^{n} p_{ij} = 1 \ (i = 1, 2, \ldots, n)$$

$$p_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j .$$

Let $\mathcal{D}$ denote the set of all $(n \times n)$ doubly stochastic matrices and $\mathcal{P}$ denote the set of all $(n \times n)$ permutation matrices. Then we have the following result.

Result 5.10.1 *The set $\mathcal{D}$ of all $(n \times n)$ doubly stochastic matrices form a convex set and permutation matrices $P \in \mathcal{P}$ are its extreme points.*

It is simple to prove that $\mathcal{D}$ is a convex set. However to prove that the permutation matrices $P \in \mathcal{P}$ are the extreme points of $\mathcal{D}$ is not that straight forward in general. Some readers may try to prove this result for $n = 2$ which is relatively easy. The following illustration may help in understanding the given result Result (5.10.1) more clearly.

Let us consider the two (2×2) matrices

$$D = \begin{pmatrix} 1/3 & 2/3 \\ 2/3 & 1/3 \end{pmatrix}$$

and

$$P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Clearly D is a doubly stochastic matrix and P_1 is a permutation matrix. Let us consider another permutation matrix

$$P_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Then $D = 2/3P_1 + 1/3P_2$, i.e. D can be written as a convex combination of permutation matrices P_1 and P_2. This follows because $\mathcal{D}$ is a convex set having finitely many extreme points, namely all $(n \times n)$ permutation matrices. Therefore every doubly stochastic matrix $D \in \mathcal{D}$ can be written as a convex combination of its extreme points, namely $P \in \mathcal{P}$. For $n = 2$, the only extreme points of $\mathcal{D}$ are P_1 and P_2.

In view of the above, we can also view the assignment problem (5.31) as an optimization problem over the set of all $(n \times n)$ permutation matrices $\mathcal{P}$, i.e. given an $(n \times n)$ cost matrix $C = (c_{ij})$ we have to find an $(n \times n)$ matrix $\overline{P} = (\overline{p}_{ij}) \in \mathcal{P}$ such that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, \overline{p}_{ij} = \underset{P \in \mathcal{P}}{Min} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, p_{ij} \right) \tag{5.32}$$

Further, because of Result 5.10.1, the optimization problem (5.32) can also be viewed as an optimization problem over the set of all $(n \times n)$ doubly stochastic matrices $\mathcal{D}$. The main argument used here is that in (5.31) we are minimizing a linear function so the optimum will be attained at an extreme point.

The end result of the above discussion is essentially the fact that in (5.31), we can replace the constraint $x_{ij} = 0$ or 1 simply by $x_{ij} \geq 0$ for all i and j. Therefore the mathematical model of the assignment problem can also be taken as

$$\text{Min} \qquad z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n)$$

$$x_{ij} \geq 0 \text{ for all } \quad i \text{ and } j\,. \tag{5.33}$$

We can also get the equivalence of problems (5.31) and (5.33) by viewing (5.31) as a transportation problem. We discuss this equivalence in the next section.

5.11 A Transportation Equivalent of the Assignment Problem

While studying the balanced transportation problem we observed that if availabilities a_i and requirements b_j are integers then its optimal solution will also be in terms of integers. Now problem (5.33) is a special case of the balanced transportation problem with $m = n$; $a_i = 1$ for all i and $b_j = 1$ for all j. Since a_i and b_j are integers, its optimal solution x^*_{ij}(as a transportation problem) will also be in integers. But $a_i = 1$ for all i and $b_j = 1$ for all j. This together with the fact that x^*_{ij} has to be a non negative integer for all i and j, implies that $x^*_{ij} = 0$ or 1. Therefore the assignment problem (5.31) is equivalent to the special transportation problem (5.33).

The above discussion suggests that a possible way to solve the assignment problem (5.31) could be to solve the equivalent transportation problem (5.33) by the usual transportation algorithm. This is illustrated by the below given example.

Example 5.11.1 *Consider the following cost minimizing assignment problem*

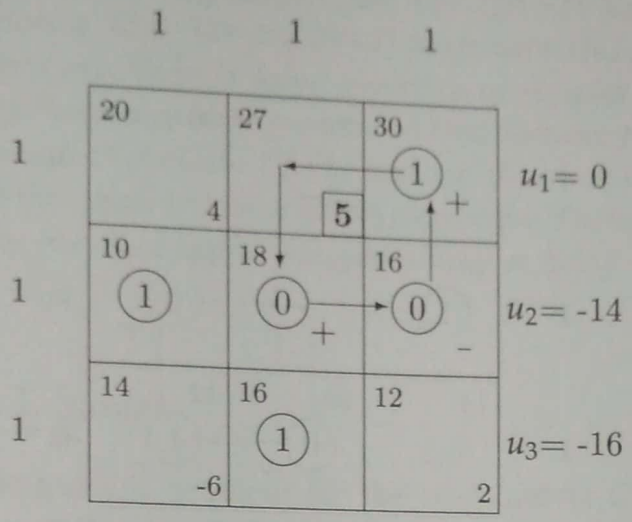
	J_1	J_2	J_3
P_1	*20*	*27*	*30*
P_2	*10*	*18*	*16*
P_3	*14*	*16*	*12*

and solve the same by solving the equivalent transportation problem.

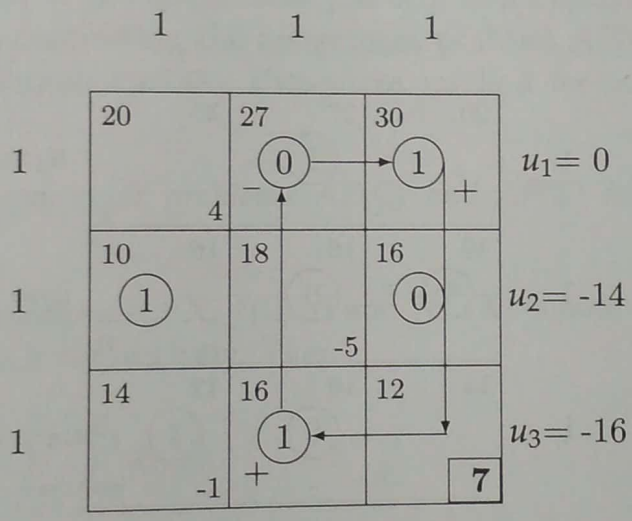
Solution Here $m = n = 3$, $a_1 = a_2 = a_3 = 1$ and $b_1 = b_2 = b_3 = 1$. Also $c_{11} = 20$, $c_{12} = 27$, $c_{13} = 30$, $c_{21} = 10$, $c_{22} = 18$, $c_{23} = 16$, $c_{31} = 14$, $c_{32} = 16$, and $c_{33} = 12$. Therefore the equivalent transportation problem to be solved is

	1	1	1
1	20	27	30
1	10	18	16
1	14	16	12

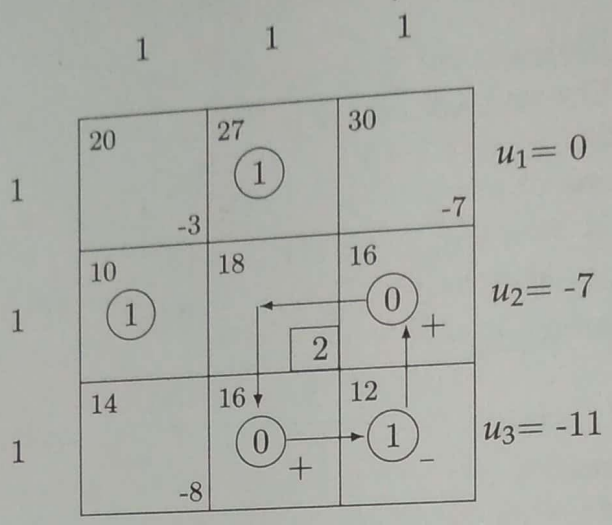
We now solve the above problem by the usual transportation algorithm by finding the starting solution by the method of column minima. This gives the following tableaus.



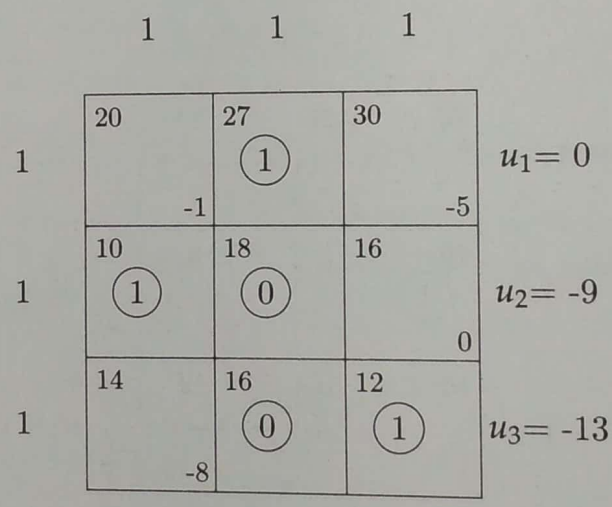
$v_1 = 24 \quad v_2 = 32 \quad v_3 = 30$



$v_1 = 24 \quad v_2 = 32 \quad v_3 = 30$



$v_1 = 17 \quad v_2 = 27 \quad v_3 = 23$



$v_1 = 19 \quad v_2 = 27 \quad v_3 = 25$

Therefore we obtain $x_{12}^* = 1 = x_{21}^* = x_{33}^*$ and other $x_{ij}^* = 0$. Since $x_{ij}^* = 1$ if the ith person is assigned the jth job, we obtain an optimal solution of the given assignment problem as $P_1 \to J_2$, $P_2 \to J_1$ and $P_3 \to J_3$. Further the minimum cost of completing all the jobs is 27+10+12 =49.

Remark 5.11.1 *Though the assignment problem (5.31) can certainly be solved by solving its equivalent transportation problem (5.33), it does not seem to be an efficient way for solving the same. The main reason being that for the equivalent transportation problem (5.33), the starting basic feasible solution as well as all subsequent basic feasible*

solutions are highly degenerate. In any tableau, we need $(2n-1)$ basic variables. But out of these $(2n-1)$ basic variables, $(2n-1)-n=(n-1)$ basic variables are always at the zero level. Such high degeneracy results in a large sequence of simplex tableaus(equivalently transportation tableaus) without any improvement in the objective function value. Therefore we need to have a relook at problem (5.31) and see if something better can be done rather than just applying the usual transportation algorithm. Fortunately this is possible and it is accomplished by the Hungarian Method which is being discussed here in the subsequent sections.

5.12 An Important Lemma

Let $AP(C)$ denote the assignment problem for the cost matrix $C=(c_{ij})$. Let $\hat{C}=(\hat{c}_{ij})$ where $\hat{c}_{ij}=c_{ij}\pm\alpha_i\pm\beta_j$, $\alpha_i\in\mathbf{R}_+$, $(i=1,2,\ldots,n)$ and $\beta_j\in\mathbf{R}_+$, $(j=1,2,\ldots,n)$. Thus the matrix $\hat{C}$ is obtained from the matrix C by adding or subtracting a fixed number from all the elements of a particular row or particular column. As the assignment problem depends upon the cost matrix only, the feasible region of both $AP(C)$ and $AP(\hat{C})$ is same. We call a feasible solution of the assignment problem as an *assignment.*

The following lemma connecting the assignment problem $AP(C)$ and $AP(\hat{C})$ plays a crucial role in the development of the Hungarian method for solving the assignment problem.

Lemma 5.12.1 *The assignment problems $AP(C)$ and $AP(\hat{C})$ have the same optimal assignments.*

Proof. Let for a feasible assignment X, $f(C,X)$ and $f(\hat{C},X)$ denote the objective function values of $AP(C)$ and $AP(\hat{C})$ respectively. Then

$$\begin{aligned} f(\hat{C},X) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\hat{c}_{ij}\,x_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}\pm\alpha_i\pm\beta_j)\,x_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}\,x_{ij}\pm\sum_{i=1}^{n}\alpha_i\left(\sum_{j=1}^{n}x_{ij}\right)\pm\sum_{j=1}^{n}\beta_j\left(\sum_{i=1}^{m}x_{ij}\right) \\ &= f(C,X)\pm\sum_{i=1}^{n}\alpha_i\pm\sum_{j=1}^{n}\beta_j\,. \end{aligned} \tag{5.34}$$

As α_i and β_j are known constants, we have

$$f(\hat{C},X)=f(C,X)+\text{constant} \tag{5.35}$$

Therefore $AP(C)$ and $AP(\hat{C})$ will have the same optimal assignments $\overline{X}$ though the objective function values $f(C, \overline{X})$ and $f(\hat{C}, \overline{X})$ will differ in general. □

Remark 5.12.1 *In view of Lemma 5.12.1, without any loss of generality we can assume that $c_{ij} \geq 0$ for all i and j. This is because if some $c_{ij} < 0$ then we can always add a positive number α to all elements of C to get a matrix $\overline{C}$ in which all $\overline{c_{ij}} \geq 0$, and then C and $\overline{C}$ will have the same optimal assignment due to Lemma 5.12.1.*

Motivation For The Hungarian Method

We now assume that for the given cost matrix C, all $c_{ij} \geq 0$, which implies that for any feasible assignment x_{ij}, the objective function value $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} \geq 0$. Therefore employing Lemma 5.12.1, if starting from C we get matrices $C_1, C_2, \ldots, C_k$ such that for the cost matrix C_k, there exists a feasible assignment x_{ij}^* such that $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k\, x_{ij}^* = 0$, then x_{ij}^* is an optimal assignment for C_k and hence for $C_{k-1}, C_{k-2}, \ldots, C_3, C_2, C_1$ and C. Here it must be understood that matrices $C_1, C_2, \ldots, C_k$ have to be obtained from C by employing the operation $c_{ij} \pm \alpha_i \pm \beta_j$ only so that Lemma 5.12.1 remains applicable.

Since we want that $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k\, x_{ij}^* = 0$ for the matrix C_k and x_{ij}^* is a feasible assignment, it makes sense to have'enough' entries as'zero' in C_k at appropriate positions. Thus our first goal should be to perform operations of the form $c_{ij} \pm \alpha_i \pm \beta_j$ on C to generate more 'zeros' so that at some stage we get a matrix C_k which has'enough' zeros, i.e. for C_k we have $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k\, x_{ij}^* = 0$. The Hungarian method for solving $AP(C)$ is a systematic procedure to accomplish this goal.

5.13 The Hungarian Method for solving the Assignment Problem

We now describe the Hungarian method for solving $AP(C)$ which is based on generating 'enough' zeros in C as discussed in Section 5.12. Later, in the next section, we shall have a more mathematical dual interpretation of the Hungarian method.

Step 1 Choose the smallest element in each row of C and subtract the same from all elements of the corresponding row. This will result in a matrix C_1 which has at least one 'zero' in every row.

Step 2 Choose the smallest element in each column of C_1 and subtract the same from all elements of the corresponding column. This will result in a matrix C_2 which has at least one 'zero' in every row and every column.

Now we wish to check if the matrix C_2 has enough 'zeros'. For this we perform Step 3 as given below but before that we give the following definition.

Definition 5.13.1 (Independent Zeros). *Let C be an $(n \times n)$ matrix which has at least one 'zero' in every row and every column. Then any two 'zeros' of C are independent if*

they do not lie in the same row or in the same column. Further, more than two 'zeros' of C are independent if every two of them are so.

Let r be the maximum number of independent zeros in C, then r is called the index of C. Obviously $r \leq n$.

Step 3 Find the maximum number of independent zeros in C_2, i.e. its index r. If $r = n$, then stop as an optimal assignment for $AP(C_2)$ has been obtained where the assignments are made at the positions of independent zeros in C_2. Since C_2 and C_1 both have been obtained by employing Lemma 5.12.1 the optimal assignment for $AP(C_2)$ will remain optimal for $AP(C_1)$ as well as for $AP(C)$. However if $r < n$, then it is an indication that C_2 still does not have 'enough' zeros and then we go to Step 4.

Before we discuss Step 4, we take two examples where in the first example Step 3 results in $r = n$, while for the second example it results in $r < n$.

Example 5.13.1 *Find an optimal assignment of persons to the jobs for the cost minimizing assignment problem AP(C) where*

$$C = \begin{pmatrix} 27 & 20 & 30 \\ 16 & 18 & 17 \\ 12 & 14 & 16 \end{pmatrix}$$

Solution Performing Steps 1 and 2 of the Hungarian method we get

$$C_1 = \begin{pmatrix} 7 & 0 & 10 \\ 0 & 2 & 1 \\ 0 & 2 & 4 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 7 & 0 & 9 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{pmatrix}$$

Now we find the maximum number of independent zeros in C_2. For this we first scan through rows of C_2 and if we find a row having only one 'zero' we make that 'zero' as an assigned zero by enclosing that in a rectangle and cross all other 'zeros' in that row and column. Continuing in this manner with the rows of C_2, we repeat the same thing with the columns of C_2. For our example we get $r = 3$ as shown below

$$C_2 = \begin{pmatrix} 7 & \boxed{0} & 9 \\ \cancel{0} & 2 & \boxed{0} \\ \boxed{0} & 2 & 3 \end{pmatrix}$$

As $r = 3 = n$, we have got an optimal assignment for $AP(C_2)$ because making assignments at the position of independent zeros gives

$$X^* = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

and $\sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}^{(2)} x_{ij}^* = 0$. Therefore optimal assignment for $AP(C_1)$ as well as for $AP(C)$ is also X^*. Thus the optimal assignment of persons to jobs is $P_1 \to J_2$, $P_2 \to J_3$ and $P_3 \to J_1$ with the minimum cost of assignment is 20+17+12=49.

Example 5.13.2 *Find an optimal assignment of persons to the jobs for the assignment problem AP(C) where*

$$C = \begin{pmatrix} 20 & 27 & 30 \\ 10 & 18 & 16 \\ 14 & 16 & 12 \end{pmatrix}.$$

Solution Performing Steps 1 and 2 of the algorithm we get

$$C_1 = \begin{pmatrix} 0 & 7 & 10 \\ 0 & 8 & 6 \\ 2 & 4 & 0 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} \boxed{0} & 3 & 10 \\ \cancel{0} & 4 & 6 \\ \cancel{2} & \boxed{0} & \cancel{0} \end{pmatrix}$$

Therefore $r = 2(< 3)$ and so have to go Step 4. Now we discuss Step 4 of the Hungarian method. As the basic aim of this step is to possibly generate more 'zero' in C_2 by employing Lemma 5.12.1, it makes sense to cover the existing 'zeros'(both assigned $\boxed{0}$ as well unassigned $\otimes$) of C_2 so that they are intact as far as possible. For this we draw the minimum number of lines to cover all 'zeros' of C_2. In this case we need only two lines to cover all 'zeros' of C_2 as shown here. It is not surprising that we need only two lines to cover all 'zeros' of C_2 and also for C_2, we have $r = 2$. This follows because of Koëing's theorem stated below.

Theorem 5.13.1 (Koëing's Theorem) *The minimum number of lines to cover all 'zeros' in C_2(in general C_k) equals the maximum number of independent zeros of C_2(in general C_k).*

After isolating the existing 'zeros' of C_2, to generate(if possible) more 'zeros' in C_2, we choose the smallest element of C_2 which is uncovered. Let this be s. For our example $s = 3$. We next subtract s from all elements of C_2 to get

$$C_2^{(1)} = \begin{pmatrix} -3 & 0 & 7 \\ -3 & 1 & 3 \\ -1 & -3 & -3 \end{pmatrix}$$

and by Lemma 5.12.1, C_2 and $C_2^{(1)}$ have the same optimal assignment. But in $C_2^{(1)}$ we have lost the 'zeros' of the first column and the third row. To recover these 'zeros' we add $s = 3$ to all elements of the first column of $C_2^{(1)}$. This gives

$$C_2^{(2)} = \begin{pmatrix} 0 & 0 & 7 \\ 0 & 1 & 3 \\ 2 & -3 & -3 \end{pmatrix}$$

In a similar manner we add $s = 3$ to all elements of the third row of $C_2^{(2)}$ to get

$$C_2^{(3)} = \begin{pmatrix} 0 & 0 & 7 \\ 0 & 1 & 3 \\ 5 & 0 & 0 \end{pmatrix}$$

If we call $C_2^{(3)}$ as C_3 then C_3 could have been obtained from C_2 directly by performing Step 4 as described below.

Step 4 (a) Find the minimum number of lines to cover all 'zeros' of C_2. By Koeing's theorem this equals the maximum number of independent 'zeros' in C_2.
(b) Choose the smallest element in C_2 which is uncovered, i.e which does not lie on any line. Let this number be s.
(c) Subtract this number s from all elements of C_2 which do not lie on any line.
(d) Add this number s to all elements of C_2 which lie on two lines.
(e) Leave other elements(i.e. those elements which lie only on one line) as such.

Step 4 will result in the matrix C_3. Here it must be noted that C_3 is infact the matrix resulting from the matrix C_2 via $C_2^{(1)}$, $C_2^{(2)}$ and $C_2^{(3)}$. Therefore C_3, C_2, C_1 and C all have the same optimal assignment.

Step 5 Now go to Step 3 replacing C_2 by C_3 and continue.

For our example, we get

$$C_3 = \begin{pmatrix} \cancel{0} & \boxed{0} & 7 \\ \boxed{0} & 1 & 3 \\ 5 & \cancel{0} & \boxed{0} \end{pmatrix}$$

with r = Index $C_3 = n = 3$. Therefore we stop and get an optimal assignment for C_3(and hence for C_2, C_1, C) at the position of independent 'zeros'. This gives the optimal assignment as $P_1 \to J_2$, $P_2 \to J_1$, $P_3 \to J_3$ with the minimum cost of assignment as 27+10+12=49.

Remark 5.13.1 *While solving Example (5.13.2), we have drawn the two lines in C_2 just by observation. This may be possible for a small problem but we need to have an algorithm to find the minimum number of lines to cover all zeros. Also we need to have a convergence proof so that we certainly have a matrix C_k(k finite) for which Index $(C_k) = n$. While we shall discuss the convergence aspects in Section 5.15, we do present an algorithm for finding the minimum number of lines as desired.*

Algorithm for Finding The Minimum Number of Lines to Cover all 'Zeros' in C_2: Details of Step 4(a)

(i) Mark (√) all rows for which assignments have not been made.
(ii) Mark (√) all columns, not already marked, which have unassigned zeros in the marked rows.
(iii) Mark (√) all rows, not already marked, which have assigned zeros in the columns.
(iv) Repeat (ii) and (iii) above till the chain of marking ends.

(v) Draw lines through all the *unmarked rows* and *marked columns*. This will give the minimum number of lines to cover all the 'zeros' of C_2.

We illustrate the above algorithm for the matrix C_2 as obtained in Example (5.13.2). We have

$$C_2 = \begin{pmatrix} \boxed{0} & 3 & 10 \\ \cancel{0} & 4 & 6 \\ \cancel{2} & \boxed{0} & \cancel{0} \end{pmatrix} \begin{matrix} \surd \\ \surd \\ \\ \end{matrix}$$

As there is no assigned zero in the second row, we mark (√) that row. Now in the mark row(i.e. the second row for our example) we look the position of unassigned zeros and mark the corresponding column. In our example, the unassigned zero is at the position (2,1) so we mark the first column. Now in the first column(i.e. the marked column) we look the position of assigned zeros and mark the corresponding rows. Here, in the first column, the assigned zero falls in the first row so we mark the first row. But there is no unassigned zero in the first row so the chain of marking has ended. Now we draw the lines through the unmarked rows and marked columns to get the minimum number of lines to cover all the zeros of C_2 as shown here.

5.14 A Dual Interpretation of the Hungarian Method

We consider the transportation equivalent of the assignment problem $AP(C)$ given at (5.31) and write its dual as

$$\text{Min} \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

subject to

$$u_i + v_j \le c_{ij} \quad \text{for all } i \text{ and } j. \tag{5.36}$$

Here u_i and v_j are the dual variables which are unrestricted in sign. Let x_{ij} be a feasible assignment of $AP(C)$ and u_i $(i = 1,2,\ldots,n)$, v_j $(j = 1,2,\ldots,n)$ be feasible for the dual problem (5.36), then for optimality

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j,$$

i.e.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} = \sum_{i=1}^{n} u_i \left(\sum_{j=1}^{n} x_{ij}\right) + \sum_{j=1}^{n} v_j \left(\sum_{j=1}^{n} x_{ij}\right),$$

i.e.

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(c_{ij}-u_i-v_j\right)x_{ij}=0 \tag{5.37}$$

But $x_{ij}\geq 0$ and $c'_{ij}=c_{ij}-u_i-v_j\geq 0$ as x_{ij} and (u_i,v_j) are feasible to (5.31) and (5.36) respectively. Therefore (5.37) implies

$$c'_{ij}\,x_{ij}=0 \text{ for all } i \text{ and } j \tag{5.38}$$

The optimality conditions (5.38) are the usual complementarity conditions which have to be verified for optimality. The Hungarian method, as discussed in Section 5.13 essentially finds feasible vectors (u_i,v_j) and $X=(x_{ij})$ such that (5.38) is satisfied. We now explain this interpretation of the Hungarian method.

1. In Step 1, we find the smallest element in each row of C and subtract the same from all elements of the corresponding row to get the matrix C_1. This essentially finds the dual variable u_i $(i=1,2,\ldots,n)$ given by

$$u_i=\underset{1\leq j\leq n}{Min}(c_{ij})$$

and then $c_{ij}^{(1)}$ equals $c_{ij}-u_i$ for all i and j.

2. In Step 2, we find the smallest element in each column of C_1 and subtract the same from all elements of the corresponding column to get the matrix C_2. This essentially finds the dual variables v_j $(j=1,2,\ldots,n)$ given by

$$v_j=\underset{1\leq i\leq n}{Min}(c_{ij}^{(1)})=\underset{1\leq i\leq n}{Min}(c_{ij}-u_i)$$

and then $c_{ij}^{(2)}$ equals $c_{ij}-u_i-v_j$ for all i and j.

Here it may be noted that $cij^{(1)}\geq 0$ and $c_{ij}-u_i-v_j\geq 0$ for all i and j. Thus at the end of Step 2, a feasible solution (u_i,v_j) of the dual of $AP(C)$ has been obtained. In Step 3, when $r<n$, we essentially update the current feasible solution (u_i,v_j) suitably so that eventually we obtain a feasible solution $(u_i^{(k)},v_j^{(k)})$ of the dual of $AP(C_k)$ for which the complementary slackness conditions (5.38) hold. At this stage we not only get the optimal assignment x_{ij}^* but also have $c_{ij}^{(k)}-u_i^{(k)}-v_j^{(k)}=0$ for all $(2n-1)$ basic variables. The updation of the current dual feasible solution (u_i,v_j) is given in Step 4 of the algorithm where the matrix C_3 is obtained. Infact various operations which have been performed in Step 4 to get the matrix C_3 are essentially the following updation rules

$$u_i^{(2)}=\begin{cases}s & \text{if } i^{th} \text{ row is uncovered(i.e. it is a marked row)}\\ 0 & \text{otherwise}\end{cases} \tag{5.39}$$

$$v_j^{(2)} = \begin{cases} -s & \text{if } j^{th} \text{ column is covered(i.e. it is a marked column)} \\ 0 & \text{otherwise} \end{cases} \tag{5.40}$$

This gives

$$c_{ij}^{(3)} = c_{ij}^{(2)} - u_i^{(2)} - v_j^{(2)} \geq 0,$$

where $u_i^{(2)}$ and $v_j^{(2)}$ are the updated u_i and v_j values as given above, and s is the smallest uncovered element in C_2.

We illustrate this with the help of the below given example.

Example 5.14.1 *Consider the assignment problem AP(C) of Example 5.13.2 and find its optimal solution. Also find the optimal solution of the dual of AP(C).*

Solution We have

$$C = \begin{pmatrix} 20 & 27 & 30 \\ 10 & 18 & 16 \\ 14 & 16 & 12 \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & 7 & 10 \\ 0 & 8 & 6 \\ 2 & 4 & 0 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} \boxed{0} & 3 & 10 \\ \cancel{0} & 4 & 6 \\ 2 & \boxed{0} & \cancel{0} \end{pmatrix}$$

(✓ ✓ ✓)

Now $s = 3$ and hence from the update rules (5.39) and (5.40) we have $u_1^{(2)} = 3$, $u_2^{(2)} = 3$, $u_3^{(2)} = 0$, $v_1^{(2)} = -3$, $v_2^{(2)} = 0$, $v_3^{(2)} = 0$ is a feasible solution of the dual of $AP(C_2)$. Therefore we can evaluate $c_{ij}^{(3)} = c_{ij}^{(2)} - u_i^{(2)} - v_j^{(2)}$ to get the matrix C_3 as

$$C_3 = \begin{pmatrix} 0 & \boxed{0} & 7 \\ \boxed{0} & 1 & 3 \\ 5 & 0 & \boxed{0} \end{pmatrix}. \tag{5.41}$$

Also looking at the position of independent zeros in C_3 we have

$$X^* = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.42}$$

But from (5.41) and (5.42) we have $c_{ij.}^{(3)}\, x_{ij}^* = 0$ for all i and j and hence the conditions (5.38) are satisfied. Therefore X^* is an optimal assignment for $AP(C)$, i.e. $P_1 \to J_2$, $P_2 \to J_1$ and $P_3 \to J_3$.

To get an optimal solution of the dual of $AP(C)$, we may consider the transportation equivalent of the given problem and solve the same by the transportation algorithm. We may refer to Example 5.11.1 in this regard where the problem of the given Example 5.14.1 has already been solved by the transportation algorithm. In the last transportation tableau, we not only get the optimal assignment but also an optimal solution of the dual problem. As obtained in Example 5.11.1, for the given assignment problem $AP(C)$, the optimal solution of its dual is $\overline{u_1} = 0$, $\overline{u_2} = -9$, $\overline{u_3} = -13$, $\overline{v_1} = 19$, $\overline{v_2} = 27$, $\overline{v_3} = 25$ and the optimal value of the dual is 49.

There is a much better way of getting an optimal solution of the dual of $AP(C)$ without resorting to the transportation algorithm from the very beginning. From (5.42) the optimal transportation tableau of $AP(C)$ is

	1	1	1
1	20	27 (1)	30
1	10 (1)	18 (0)	16
1	14	16 (0)	12 (1)

which gives $u_1 + v_2 = 27$, $u_2 + v_1 = 10$, $u_2 + v_2 = 18$, $u_3 + v_2 = 16$, $u_3 + v_3 = 12$. Taking $u_1 = 0$ we get $u_2 = -9$, $u_3 = -13$, $v_1 = 19$, $v_2 = 27$ and $v_3 = 25$ as an optimal solution of the dual of AP(C).

In this context it may be noted that the optimal solution of the dual of $AP(C)$ is not unique. This is because there are alternate possibilities of putting the 'zero' basic cells in the tableau and choosing arbitrary value of one of the variables u_i and v_j.

Remark 5.14.1 *While using the Hungarian method, it is not true that with each iteration the number of 'assigned zeros' increases by one. For example, if we consider problem AP(C) for*

$$C = \begin{bmatrix} 14 & 4 & 10 & 16 & 8 & 2 \\ 21 & 6 & 15 & 24 & 12 & 3 \\ 35 & 10 & 25 & 40 & 20 & 5 \\ 56 & 16 & 40 & 64 & 32 & 8 \\ 42 & 12 & 30 & 48 & 24 & 6 \\ 28 & 8 & 20 & 32 & 16 & 4 \end{bmatrix}$$

then we can verify that the maximum number of independent zeros in C_2 is 2. But the maximum number of independent zeros in C_3, C_4, and C_5 remains 3. Further it remains 4 for C_6 and C_7, and the optimal assignment, i.e. six independent zeros are obtained only for C_{12}. The only guarantee of the method is that the maximum number of independent zeros will not decrease and for some finite k, $AP(C_k)$ will have n independent zeros.

5.15 Finite Convergence of the Hungarian Method

In this section we prove the finite convergence of the Hungarian method. Let us recollect that for the assignment problem $AP(C)$ with $C = (c_{ij})_{n\times n}$, Steps 1 and 2 of the algorithm give the matrix $C_2 = (c_{ij}^{(2)})$ where for all i and j, $c_{ij}^{(2)} = (c_{ij} - u_i - v_j) \geq 0$.

Further, in case $r < n$, Step 4 of the algorithm generates the matrix $C_3 = (c_{ij}^{(3)})$, $c_{ij}^{(3)} = (c_{ij}^{(2)} - u_i^{(2)} - v_j^{(2)})$, where $u_i^{(2)}$ and $v_j^{(2)}$ are updated dual variables as per equations (5.39) and (5.40). Also $c_{ij}^{(3)} = c_{ij}^{(2)} - u_i^{(2)} - v_j^{(2)} \geq 0$ for all i and j. Now from equations (5.39) and (5.40) we have

$$c_{ij}^{(2)} - c_{ij}^{(3)} = \begin{cases} s & \text{if row } i \text{ and column } j \text{ both are unmarked} \\ -s & \text{if row } i \text{ and column } j \text{ both are marked} \\ 0 & \text{if row } i \text{ is marked and column } j \text{ is not marked} \\ & \text{or if row } i \text{ is unmarked and column } j \text{ is marked} \end{cases}$$

Therefore

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(2)} - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(3)} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(c_{ij}^{(2)} - c_{ij}^{(3)}\right)$$
$$= p(n-q)s - (n-p)qs$$
$$= n(p-q)s, \tag{5.43}$$

where p is the number of marked rows and q is the number of marked columns.

We can verify (5.43) for Example 5.14.1, where $n = 3$, $s = 3$, $p = 2$, and $q = 1$. For this example

$$C_2 - C_3 = \begin{pmatrix} 0 & 3 & 3 \\ 0 & 3 & 3 \\ -3 & 0 & 0 \end{pmatrix}$$

Therefore $\sum_{i=1}^{3}\sum_{j=1}^{3}\left(c_{ij}^{(2)} - c_{ij}^{(3)}\right) = 12 - 3 = 9$ which equals $n(p-q)s = 3(2-1)3 = 9$.

Let r denote the maximum number of independent 'zeros' in C_2. Then by Koëing's theorem r equals the minimum number of lines to cover all 'zeros' in C_2. But this minimum number equals the sum of the number of unmarked rows and the number of marked columns, i.e. $r = (n-p)+q$, or equivalently $(p-q) = n-r$. Therefore (5.43) gives

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(c_{ij}^{(2)} - c_{ij}^{(3)}\right) = n(p-q)s = n(n-r)s\,. \tag{5.44}$$

Now for $r < n$ i.e. when the current assignment is not optimal, $n(n-r)\,s > 0$. Also without any loss of generality we can assume that C has integer entries(when c_{ij} are rational, we can always take L.C.M), which gives that s is an integer. Therefore $n(n-r)\,s$ is an integer.

The above discussion shows that at the end of Step 4 we have (i) a new matrix is made available on which markings can be continued (ii) the sum of all the entries of the new matrix (here C_3) is reduced by a positive integer $n(n-r)\,s$ and (iii) all the entries of the matrix C_3 are non-negative.

Hence the Hungarian method is bound to terminate in finite number of iterations as $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(k)}$ can never be negative.

5.16 Summary and Additional Notes

- Section 5.2 describes the transportation problem and gives its mathematical model.

- Sections 5.3-5.8 present the complete working of the transportation algorithm for solving the balanced TP. The importance of unimodularity and related results are discussed in Section 5.4.
- The unbalanced transportation problem is discussed in Section 5.9.
- Section 5.10 discusses the assignment problem and gives its mathematical model. The transportation equivalent of the assignment problem is given in Section 5.11.
- Sections 5.12-5.14 present the complete working of the Hungarian method for solving the assignment problem while Section 5.15 establishes its finite convergence.
- The transportation problem was first formulated by F.L. Hitchcock in 1941 which was later discussed in detail by T.C. Koopman in 1949. L.V. Kantrovitch also discussed the transportation problem in 1951.
- The adaptation of the simplex method to solve TP was given by G.B. Dantzig in 1954 and also by A.Charnes and W.W.Cooper in 1954.
- There is a very close connection between the transportation and assignment problems, and certain bipartite graphs. For a graph theoretic treatment of the transportation and assignment problem we may refer to the texts by Berge and Ghouila-Houri [18] and Bazaraa et al. [12].
- The Hungarian method for solving the assignment problem was developed by H.W. Kuhn in 1955.
- Similar to the cost minimizing transportation and assignment problems, we may also study time minimizing problems. These problems have been studied in the literature e.g. Hammer [74], Garfinkel and Rao [64], and Swarc [152].
- A very natural extension of the transportation problem is the multi-commodity transportation problem; originally studied by Haley [73], who also gave many variants of the same. These problems have many applications in mining industries.
- Burkard [30] studied the cost minimizing TP and the time minimizing TP in an unified setting of *algebraic linear programming*. He developed a general Hungarian method for the algebraic transportation problem which is applicable to both the cost minimizing as well as the time minimizing transportation problem. Here the transportation problem over a commutative ordered semigroup is defined which subsumes both types (cost and time) of transportation problems as special cases. Frieze [59] studied the algebraic assignment problems in the setting of algebraic linear programming.
- If we allow that an item may reach a destination via another source or via another destination or a combination of these, then the transportation problem is called the transportation with transhipment or in short the *transhipment problem*. In this chapter we have only studied the transportation problem without transhipment. The transhipment problem (with m sources and n destination) can be transformed into a transportation problem (without transhipment) with $(m+n)$ sources and $(m+n)$ destination and then can be solved by the usual algorithm. We may refer to Taha [154] in this regard.

- The assignment problem is also related with the famous *travelling salesperson problem (TSP)*. In fact solving an n city, travelling salesperson problem is equivalent to finding a *cyclic solution* of the usual n-person -n-jobs assignment problem. In the literature, there are certain approximate algorithms for solving TSP based on this equivalence.

5.17 Exercises

5.1 *Consider the following (TP)*

	40	50	10
40	2	5	2
30	4	1	3
30	3	6	2

Determine a starting b.f.s by employing

1. *the north west corner rule*
2. *the column minima*
3. *the row minima*
4. *the matrix minima and*
5. *VAM(Vogel's Approximation Method).*

5.2 *Solve the following cost minimizing transportation problems*

(i)

	44	13	15
24	2	1	3
14	4	6	5
34	8	9	7

(ii)

	25	10	20
20	5	1	0
10	3	2	4
15	7	5	2
15	9	6	1

5.3 *Solve the following TP and identify the source which will have excess supply*

	45	10	15
20	5	1	0
20	3	2	4
15	7	5	2
25	8	6	0

5.4 *In an unbalanced transportation problem, sometimes there are penalties for unsatisfied demand, to reflect the failure of the supplier to meet the required demand. Consider the problem*

	60	50	50
90	1	5	6
10	3	2	3
20	2	6	1

Let the penalty costs per unit of unsatisfied demand be 6,4,and 2 respectively for destinations D_1, D_2 and D_3. Find an optimal solution of the given (TP). Also identify the destinations which will have short supply and by how many units.

5.5 *Consider the transportation problem given at Question 5.4 above. Suppose it is required that the demand at the destination D_1 must be satisfied exactly. Suggest a procedure to solve the given transportation problem and also identify the destination which will have maximum short supply.*

5.6 *Solve the following (TP) by finding the initial solution using the method of column minima*

	15	18	14
10	3	6	4
15	2	3	1
30	5	2	

Here the destination D_3 has black listed source S_3 and will not accept any supply from that source.

5.7 *Solve the following cost minimizing transportation problem starting with the b.f.s* $x_{12} = 30$, $x_{21} = 40$, $x_{32} = 20$, $x_{43} = 60$ *and other* $x_{ij} = 0$.

	40	50	60
30	4	5	2
40	4	1	3
20	3	6	2
60	2	3	7

5.8 *Consider the following transportation problem.*

	40	50	10
40	2	5	2
30	4	1	3
30	3	6	2

1. *Using the column minima method to find a starting solution, obtain the minimum cost transportation schedule.*
2. *Write the dual(D) of the given (TP) and obtain its optimal solution.*

5.9 *Consider the following transportation problem*

	1	3	4
3	2	1	1
5	1	4	4

1. *Prove by duality theory that* $(x_{11}^* = 0,\ x_{12}^* = 3,\ x_{13}^* = 0,\ x_{21}^* = 1,\ x_{22}^* = 0, x_{23}^* = 4)$ *is an optimal solution*
2. *Will this optimal solution change if each cost element is multiplied by 10?*

5.10 *1. Solve the following LPP*

Min $3x_1 + 4x_2 + 2x_4 + 3x_5 + 4x_6$

subject to

$$x_1 + x_2 + x_3 = 13$$
$$x_4 + x_5 + x_6 = 5$$
$$x_1 + x_4 = 8$$
$$x_2 + x_5 = 4$$
$$x_3 + x_6 = 6$$
$$x_1,\ x_2,\ x_3\ x_4,\ x_5,\ x_6 \geq 0.$$

2. Write the dual (D) of above LPP and obtain its optimal solution.

5.11 *Consider the following assignment problem(AP)*

	J_1	J_2	J_3	J_4
P_1	8	26	17	11
P_2	12	28	4	26
P_3	18	19	18	5
P_4	18	16	24	10

1. Solve the above problem to find the minimum cost assignment.

2. What will be the optimal assignment if it is given that Job J_4 *can not be assigned the person* P_1.

3. Find the optimal assignment if the person P_2 *is sure to do the job* J_4.

5.12 *Use the Hungarian method to solve the following (AP)*

	I	II	III	IV	V	VI
1	5	0	-6	8	7	-4
2	-5	2	-3	0	6	-6
3	3	-4	4	3	-5	2
4	3	4	9	7	-2	3
5	0	-1	-3	2	-1	2
6	4	3	2	-1	0	4

5.13 *Given the following data, find the optimal assignment of territories to salespersons so that the total sales are minimized.*

Territory	Annual Sales(in Rs)
I	60,000
II	50,000
III	40,000
IV	30,000

Also working under the same conditions, the yearly sales of persons are in the following proportions

Salesperson	Proportion
A	7
B	5
C	5
D	4

5.14 *Consider the following problem (AP)*

	J_1	J_2	J_3
P_1	20	27	30
P_2	10	18	16
P_3	14	16	12

1. *Express the above (AP) as a (LPP) and write its dual (D).*
2. *Treat the given (AP) as a special case of (TP) and solve the same by the transportation algorithm.*
3. *Solve the given (AP) by the Hungarian method and hence find an optimal solution of its dual (D).*
4. *Use the answer obtained at (ii) above to find a solution of the dual (D).*

5.15 *A company has 4 engineers to design 4 projects. Five projects are offered to the company with the following expected profit(in thousand of Rupees)*

	P_1	P_2	P_3	P_4	P_5
E_1	62	78	50	101	82
E_2	71	84	61	73	59
E_3	87	92	111	71	81
E_4	48	64	87	77	80

Due to organizational constraints, only one engineer can be assigned to a project and vice-versa. Project P_1 is very important and it must be taken by the company. Find the optimal assignment of engineers to the projects so as to maximize the total profit. Also identify the project which the company should decline.

5.16 *Consider the following LPP*

$$\text{Max} \quad 2x_1 + 3x_2 + 4x_3 + x_4 + 7x_5 + 5x_6$$

subject to

$$x_1 + x_2 \leq 1$$
$$x_3 + x_4 \leq 1$$
$$x_5 + x_6 \leq 1$$
$$x_1 + x_3 + x_5 = 1$$
$$x_2 + x_4 + x_6 = 1$$
$$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6 \geq 0.$$

1. *Solve the above LPP.*
2. *Write the dual of above LPP and solve the same.*

5.17 *Are the following statements true? Give reasons for your answer.*

1. *For a balanced transportation problem with 4 sources and 3 destinations, the columns $p_{11}, p_{13}, p_{22}, p_{23}, p_{42}$ and p_{43} of the coefficient matrix A are linearly independent.*
2. *If all the cost elements c_{ij} of an (4×4) assignment problem (AP) are increased by 5, then the optimal value will also increase by 5.*
3. *The set of all $(n \times n)$ permutation matrices is a convex set.*
4. *A balanced transportation problem with all $c_{ij} \geq 0$ can not have unbounded solution.*
5. *If A is unimodular matrix then so is A^T.*
6. *All matrices having entries as $0, +1$ and -1 are unimodular.*
7. *The vector $p_{32} - p_{22} + p_{43}$ has exactly two entries as 'one'.*
8. *Let $\overline{X} = (\bar{x}_{ij})$ be an optimal assignment of AP(C) then it is also optimal for problem $AP(C^T)$.*

5.18 *Construct an example of each of the following (if no such example is possible, then give reasons for the same).*

1. *a (5×6) matrix A which is unimodular.*
2. *a (6×9) matrix of rank 5.*
3. *a (3×3) matrix with entries as $0, 1$ and -1, which is not unimodular.*
4. *a (3×3) doubly stochastic matrix which is not a permutation matrix.*

5. *A convex set in* $\mathbf{R}^{2\times 2}$ *whose extreme points are* $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ *and* $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.
6. *A transportation problem having infinitely many optimal solutions.*
7. *An assignment problem having infinitely many optimal solutions.*
8. *A nonsingular unimodular matrix* A *for which* A^{-1} *is not unimodular.*

6
Integer Linear Programming

6.1 Introduction

Certain linear programming problems require that some or all the variables take on only integer values. We call such problems as *integer linear programming problems* (ILP's). Integer LPP's are very common in real life applications, e.g. in a production problem, the items being produced may be in complete units (say items are T.V. sets of 21″ and 29″) and therefore fractional number of items may not have any meaning. Also sometimes, because of the physical restrictions, the constraints of the problem could be 'either - or' or 'at least k out of the given m constraints' which again lead to ILP's. A special type of ILP, namely $0-1$ ILP, occurs very frequently in the area of electrical circuits.

Integer linear programming problems, inherently being combinatorial, constitute a 'hard' class of optimization problems. Our first aim in this chapter is to understand the difficulties which arise in the algorithmic development of ILP's and then to discuss two basic algorithms for solving ILP's, namely *Gomory's cutting plane method* and *Land and Doig's branch and bound method.* Though there are several modifications of these two basic methods (so we have a class of 'cutting plane methods' and 'branch and bound methods') we do not attempt to discuss them here. In literature, the theory of finite abelian groups has also been applied for solving ILP's but that again has been left out in our presentation.

6.2 Mathematical Model and Some Possible Approaches

The mathematical model of an integer linear programming problem (ILP) may be described as

$$\text{Max} \quad z = \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ \ (i = 1, \ldots, m)$$
$$x_j \geq 0 \ \ (j = 1, \ldots, n) \tag{6.1}$$

and

$$x_j \text{ integer for } J_1 \subset J,$$

where $J = \{1, 2, \ldots, n\}$.

An ILP is called an *all integer LPP* (AILP) if all the variables of the problem are constrained to take integer values only. The problem is called *mixed integer LPP* (MILP) if some but not all variables are constrained to take integer values. Thus, in (6.1), if $J_1 = J$, then the problem is all integer, otherwise it is a mixed integer LPP. Here it may be observed that as the constraints in (6.1) are given in $Ax = b$ form, the problem is all integer if all variables, including the slack and surplus variables, are constrained to take integer values. Thus the problem

$$\text{Max} \quad z = 4x_1 + 3x_2$$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 10 \\ x_1, x_2 &\geq 0 \\ x_1 \text{ and } x_2 &\text{ integer} \end{aligned} \tag{6.2}$$

is an all integer LPP, because once x_1 and x_2 are non-negative integers, the slack variables $x_3 = 8 - x_1 - x_2$ and $x_4 = 10 - 2x_1 - x_2$ are also non-negative integers. Further if in (6.2), we take the second constraint as $\sqrt{2}x_1 + x_2 \leq 10$ (instead of $2x_1 + x_2 \leq 10$), then this problem is no more all integer. It becomes a mixed integer LPP because $x_4 = 10 - \sqrt{2}x_1 - x_2$ will not meet integer requirement even if x_1 and x_2 are integers.

For the integer LPP (6.1), we define its *associated LPP*, namely the LPP obtained from the ILP (6.1) when integer constraints 'x_j integer for $j \in J_1 \subset J$' are ignored. Thus the associated LPP (say $(LP)_1$) of the given ILP (6.1) is

$$\text{Max} \quad z = \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ \ (i = 1, \ldots, m)$$
$$x_j \geq 0 \ \ (j = 1, \ldots, n). \tag{6.3}$$

The most natural and common sense approach of solving the given ILP (6.1) seems to be to solve its associated LPP as given in (6.3), by the simplex algorithm and then round off the optimal solution so obtained to the nearest integers. The example given below brings out an important fact that *rounding off is not a correct approach to solve ILP's* because this, in general, will lead to infeasibility and/or non optimality.

Let us consider the ILP given by

$$\text{Max} \quad z = 21x_1 + 11x_2$$

subject to

$$\begin{aligned} 7x_1 + 4x_2 &\leq 13 \\ x_1, x_2 &\geq 0 \\ x_1 \text{ and } x_2 &\text{ integer.} \end{aligned} \tag{6.4}$$

The feasible set of the above ILP is the following set of six discrete points

$$\{(0,0),\ (0,1),\ (1,0),\ (1,1),\ (0,2),\ (0,3)\}.$$

These points are essentially the points having integer coordinates which are inside the feasible region of the associated LPP. This we can visualize in Fig 6.1.

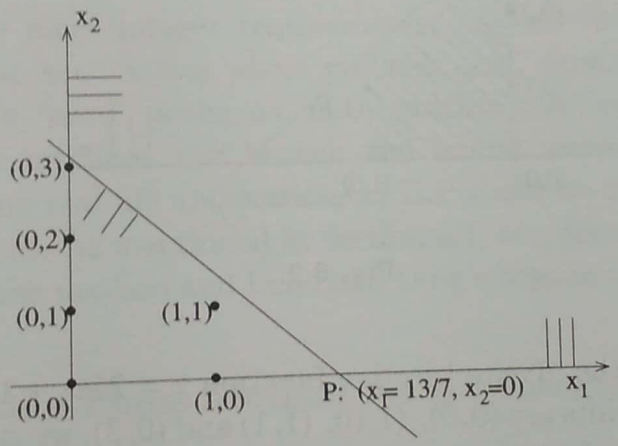


Fig. 6.1.

Now to find an optimal solution of ILP (6.4), we have to simply evaluate the value of z at each of the six points which are feasible. This gives the optimal solution of ILP (6.4) as $(x_1^\star = 0,\ x_2^\star = 3)$ and the optimal value as $z^\star = 33$.

Suppose we now solve the associated LPP of the ILP (6.4) by the simplex algorithm, i.e. solve the following

$$\text{Max} \quad z = 21x_1 + 11x_2$$

subject to

$$\begin{aligned} 7x_1 + 4x_2 &\leq 13 \\ x_1, x_2 &\geq 0. \end{aligned}$$

It is obvious that the optimal solution of the above LPP is ($\bar{x}_1 = 13/7$, $\bar{x}_2 = 0$) which is shown as point P in Fig 6.1. The rounding off of this solution, namely ($\bar{x}_1 = 13/7$, $\bar{x}_2 = 0$) gives ($\hat{x}_1 = 2$, $\hat{x}_2 = 0$) which is not even feasible for the given ILP (6.4). Further the rounded off solution ($\hat{x}_1 = 2$, $\hat{x}_2 = 0$) is 'far away' from the actual optimal solution ($x_1^\star = 0$, $x_2^\star = 3$), and the two objective function values $z(\hat{x}) = 42$ and $z(x^\star) = 33$ are not 'close' in any meaningful sense. Therefore 'rounding off' is not a correct approach to solve ILP's.

Having rejected the idea of rounding off, the next best thing seems to be to take the convex hull of the feasible set of the given ILP. This makes sense because though the feasible set of the given ILP is non convex, its convex hull is a polytope whose corner points meet the integer requirements. Therefore, in particular, if the problem is all integer then this polytope has corner points having integer coordinates only. For our example here, we show the polytope in Fig 6.2.

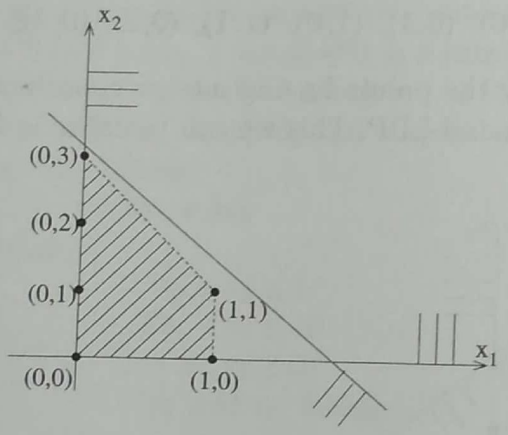


Fig. 6.2.

Now if we maximize the given objective function $z = 21x_1 + 11x_2$ over the polytope (Fig 6.2) whose corner points are $(0,0)$, $(1,0)$, $(1,1)$ and $(0,3)$, we get the optimal solution as the corner point $(0,3)$ which is also optimal for the given ILP. In other words, we are getting the optimal solution of ILP by solving the linear programming problem

$$\begin{aligned} \text{Max} \quad & z = 21x_1 + 11x_2 \\ \text{subject to} \quad & \\ & 2x_1 + x_2 \le 3 \\ & x_1 \le 1 \\ & x_1 \ge 0, x_2 \ge 0, \end{aligned}$$

where the feasible region of the LPP (6.2) is actually the convex hull of the feasible set of the given ILP (6.4).

In view of the above, we can think of solving the given ILP by solving the linear programming problem whose objective function is the same as that of the given ILP

but whose feasible region is the polytope obtained by taking the convex hull of the feasible set of the given ILP. Thus solving ILP (6.1) is equivalent to solving LPP

$$\begin{aligned} \text{Max} \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \\ & (x_1, \ldots, x_n) \in S, \end{aligned}$$

where S is the polytope obtained by taking the convex hull of those points $(x_1, \ldots, x_n)$ for which the following hold

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} x_j &= b_i \ (i = 1, \ldots, m) \\ x_j &\geq 0 \ (j = 1, \ldots, n) \\ x_j \text{ integer} \quad &\text{for } j \in J_1 \subset J = \{1, \ldots, n\}. \end{aligned}$$

This method of solving ILP's is perfectly valid except that there are certain practical difficulties in getting the desired convex hull, particularly when the dimension of the Euclidean space is more than two or three. Therefore, we again do not pursue this approach for solving ILP's. However, the observation that there is always a polytope inside the feasible region of the associated LPP of the given ILP such that the corner points of this polytope meet integer requirements, has led to two other approaches for solving ILP's. These are *cutting plane methods* and, *branch and bound methods.* As ILP's are essentially 'hard' problems, no algorithm can really be very 'efficient'. However, cutting plane methods and branch and bound methods have proved to be very satisfactory in many real life applications of ILP's and we propose to discuss them in the subsequent sections. As mentioned in Section 6.1, our discussion will be restricted to Gomory's cutting plane method and Land and Doig's branch and bound method only.

6.3 Gomory's Cutting Plane Method for All Integer Linear Programming

We consider the all integer LPP

$$\begin{aligned} \text{Max} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0 \\ & x \text{ integer}, \end{aligned} \tag{6.5}$$

and develop the most basic cutting plane method, namely the Gomory's cutting plane method for the same. Here, without any loss of generality we can assume that all entries in A, b and c are integers, because if they are rational we can always take the

LCM and get the entries as integers only. Therefore for AILP, the objective function is automatically constrained to be integer.

Let $(LP)_1$ be the associated LPP of the given AILP (6.5). Let us solve $(LP)_1$ by the simplex method and let $x^{(1)}$ be its optimal solution. If $x^{(1)}$ meets the integer requirements, i.e. all components of $x^{(1)}$ are integers, then obviously $x^{(1)}$ is optimal for the given AILP. However, if some component of $x^{(1)}$ is not an integer, then it is not optimal for the given AILP. In this situation, Gomory proposed to introduce a cut constraint, i.e. a constraint of the form $p^T x \leq d$ and append that to problem $(LP)_1$ to get a new LPP, say $(LP)_2$. The basic purpose of the cut constraint is to delete a part of the feasible region S_1, (S_1 being the feasible region of $(LP)_1$), but making sure that we do not delete any point of S_1 which has integer coordinates. Thus a cut constraint, by definition, certainly deletes some portion of the feasible region of the associated LPP, but takes care that we do not delete those points which are useful to us, namely the points having integer coordinates. Now we may solve $(LP)_2$ and repeat the procedure. Gomory derived an appropriate cut constraint and established that only *finitely many* cut constraints will be needed to solve any instance of the given AILP. The cut constraint given by Gomory is called the *Gomory's cut constraint* and this method of solving AILP is called the *Gomory's cutting plane method for all integer linear programming*. There are many other cutting plane methods available in the literature but they are all based on this basic method due to Ralph E. Gomory.

Before we provide the derivation of the *Gomory's cut constraint* for AILP, we make an important observation. The LPP's, $(LP)_1$ and $(LP)_2$ differ only by one additional constraint. Therefore $(LP)_2$ need not be solved from the very beginning. We can use the *dual simplex method* to solve it by appending the additional constraint, namely the cut constraint, to the last (optimal) tableau of $(LP)_1$. This is of course true for any two consecutive LPP's, say $(LP)_k$ and $(LP)_{k+1}$.

Derivation of the Gomory's Cut Constraint

To derive the Gomory's cut constraint, we first obtain the *canonical representation* of the LPP

$$\begin{aligned} \text{Max} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0, \end{aligned} \tag{6.6}$$

for a given feasible basis matrix B. For this, we partition A, x and c as

$$A = (B : R),\ x = col(x_B, x_R),\ c = col(c_B, c_R)$$

where the symbols have their usual meanings. This allows us to write the system $Ax = b$ as

$$(B : R)\begin{pmatrix} x_B \\ x_R \end{pmatrix} = b$$

i.e.

$$Bx_B + Rx_R = b$$

i.e.

$$x_B = B^{-1}b - B^{-1}Rx_R$$

i.e.

$$x_{B_i} = y_{i0} - \sum_{j \in R} y_{ij}x_j \quad (i = 1, \ldots, m)\,. \tag{6.7}$$

Here y_{i0} is the ith component of the vector $B^{-1}b$, y_{ij} is the ith component of the vector $y^{(j)} = B^{-1}a^{(j)}$ and $j \in R$ is understood as the index j running over the index set of nonbasic variables.

The m equations given at (6.7) are said to be the canonical representation of the system $Ax = b$, because each equation of (6.7) has only one basic variable with coefficient as 'one', and all other variables in the equation are nonbasic variables. What we have done for the constraints $Ax = b$, can also be done for the objective function, because

$$\begin{aligned} z &= c^T x \\ &= c_B^T x_B + c_R^T x_R \\ &= c_B^T(B^{-1}b - B^{-1}Rx_R) + c_R^T x_R \\ &= c_B^T(B^{-1}b) - (c_B^T B^{-1}R - c_R^T)x_R, \end{aligned}$$

which can be written as

$$x_{B_0} = y_{00} - \sum_{j \in R} y_{0j}x_j, \tag{6.8}$$

where $x_{B_0} = z$, $y_{00} = c_B^T(B^{-1}b)$ and $y_{0j} = z_j - c_j$. Therefore combining (6.7) and (6.8) we get

$$x_{B_i} = y_{i0} - \sum_{j \in R} y_{ij}x_j, \tag{6.9}$$

for $i = 0, 1, \ldots, m$, where $i = 0$ refers to the objective function and $i = 1, \ldots, m$ refer to the m constraints. Further the current b.f.s. and the current objective function value are obtained by taking $x_j = 0$ for $j \in R$ in (6.9). Thus y_{00} is the current objective function value and y_{i0} $(i = 1, \ldots, m)$ is the current b.f.s.

Since equation (6.9) holds for any feasible solution of LPP (6.6), it holds for all integer feasible points of AILP (6.5) as well because LPP (6.6) is essentially the associated LPP of AILP (6.5). Now if for any real number a we define its *fractional part* f_a as $f_a = a - [a]$,

where $[a]$ is the greatest integer function, then we note that $0 \le f_a < 1$. (Some readers may like to verify that for $a = -1$, $f_a = 0$ but for $a = -1.6$, $f_a = -1.6 - [-1.6] = -1.6 - (-2) = 2 - 1.6 = 0.4$ and not 0.6).

Now for each $i = 0, 1, \ldots, m$, (6.9) can be rewritten as

$$\sum_{j \in R} [y_{ij}] x_j + \sum_{j \in R} (y_{ij} - [y_{ij}]) x_j + x_{B_i} = [y_{i0}] + (y_{i0} - [y_{i0}]),$$

i.e.

$$\sum_{j \in R} [y_{ij}] x_j + x_{B_i} - [y_{i0}] = (y_{i0} - [y_{i0}]) - \sum_{j \in R} (y_{ij} - [y_{ij}]) x_j,$$

i.e.

$$\sum_{j \in R} [y_{ij}] x_j + x_{B_i} - [y_{i0}] = f_{i0} - \sum_{j \in R} f_{ij} x_j. \tag{6.10}$$

But as (6.10) holds for all feasible points of the LPP (6.5) and so for all integer feasible points of the given AILP, the L.H.S. of (6.10) is an integer. Therefore the R.H.S. of (6.10) must also be an integer. This gives that for each $i = 0, 1, \ldots, m$,

$$f_{i0} - \sum_{j \in R} f_{ij} x_j$$

must be an integer.

In the above $i = 0$ is also included because for the AILP, the objective function is also constrained to be integer as explained earlier.

Now $f_{ij} \ge 0$ and $x_j \ge 0$ for $j \in R$. Therefore

$$\sum_{j \in R} f_{ij} x_j \ge 0. \tag{6.11}$$

Also $f_{i0} < 1$ and so (6.11) gives

$$f_{i0} - \sum_{j \in R} f_{ij} x_j < 1. \tag{6.12}$$

Thus $f_{i0} - \sum_{j \in R} f_{ij} x_j$ is an integer which is less than one, and therefore

$$f_{i0} - \sum_{j \in R} f_{ij} x_j \le 0. \tag{6.13}$$

The inequality (6.13) is satisfied by every integer feasible point of the given AILP (6.5). We now assume that the current b.f.s. x_B is not an integer, i.e. it does not meet the integer requirements of AILP (6.5). In that case, we note that the inequality (6.13) is

not satisfied by the point x_B for some i. This is because for some i, x_{B_i} is not integer with $f_{i0} > 0$. But for the current b.f.s. x_B, (6.13) gives $f_{i0} \leq 0$, which is a contradiction. Thus the constraint (6.13) is satisfied by every integer feasible point of AILP but certainly not satisfied at least by the current b.f.s. x_B. In other words it certainly deletes a part of the feasible region of the associated LPP (at least the current b.f.s. x_B and may be many more points) but does not delete any feasible point with integer coordinates. Hence (6.13) is a valid cut constraint and it is called the *Gomory's cut constraint.*

We can also write the Gomory's cut constraint (6.13) as

$$-f_{i0} = s_i - \sum_{j \in R} f_{ij} x_j \tag{6.14}$$

and append the same to the associated LPP, $(LP)_1$, to get the new LPP, $(LP)_2$. Therefore we solve $(LP)_2$ and repeat the procedure.

Stepwise Description

The stepwise description of the Gomory's cutting plane method for AILP is as follows

Step 1 Solve the associated LPP, say $(LP)_1$, by the simplex method. Set $k = 1$.

Step 2 If the optimal solution obtained at Step 1 is integer, stop as an optimal solution of the given AILP is at hand. Otherwise go to Step 3.

Step 3 For any updated constraint i whose y_{i0} value is fractional (including $i = 0$, i.e. the objective function), generate the Gomory's cut constraint as given at (6.13). A common procedure here is to select that value of i, $0 \leq i \leq m$, for which f_{i0} value is maximum. Theoretically we can choose any i for which $f_{i0} > 0$ but the maximum of f_{i0} is chosen with the *hope* that it may give a *deeper cut.*

Step 4 Append the Gomory's cut constraint derived at Step 3 above to $(LP)_k$ to get the new LPP $(LP)_{k+1}$. Solve $(LP)_{k+1}$ by the dual simplex method and return to Step 2.

Theorem 6.3.1. *The number of Gomory's cut constraints needed to solve any instance of all integer linear programming problem is always finite.*

Though we shall not prove the above theorem, there are few points to be noted here. As the number of cut constraints needed is always finite, we are solving only finitely many LPP's to get an optimal solution of the given AILP. But, unfortunately, even for a problem of 'average' size, the number of cut constraints needed may be 'too many' (theoretically 'exponential' in the worst case) as AILP belongs to the class of 'hard' problems.

We also need to appreciate some other computational difficulties in this regard. As soon as we add a cut constraint to $(LP)_k$, we immediately introduce one additional row and one additional column in the existing simplex tableau. Since in a typical situation, the number of cut constraints needed may be very large, after some iterations the simplex

tableau may become too big to manage. There are some methods available in literature which can be augmented to take care of the problem of tableau management.

Example 6.3.2. Consider the integer LPP

$$\begin{aligned} \text{Max} \quad & z = 5x_1 + 2x_2 \\ \text{subject to} \quad & \\ & 2x_1 + 2x_2 \leq 9 \\ & 3x_1 + x_2 \leq 11 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integer.} \end{aligned} \tag{6.15}$$

Show that problem (6.15) is an AILP and hence solve the same by the Gomory's cutting plane method. Also verify graphically that each of the Gomory cut constraint is really deleting a part of the feasible region but not deleting any of the integer feasible points.

Solution The given ILP is equivalent to

$$\begin{aligned} \textit{Max} \quad & z = 5x_1 + 2x_2 + 0x_3 + 0x_4 \\ \text{subject to} \quad & \\ & 2x_1 + 2x_2 + x_3 = 9 \\ & 3x_1 + x_2 + x_4 = 11 \\ & x_1, x_2, x_3, x_4 \geq 0 \text{ and all integer.} \end{aligned} \tag{6.16}$$

As x_1, x_2 are constrained to be integers, it is assured that $x_3 = 9 - 2x_1 - 2x_2$ and $x_4 = 11 - 3x_1 - x_2$ are also integers. Therefore the ILP (6.15) is in fact AILP. We now employ the Gomory's cutting plane method to solve AILP (6.16).

First iteration

We consider the associated LPP (i.e. $(LP)_1$) given by

$$\begin{aligned} \text{Max} \quad & z = 5x_1 + 2x_2 + 0x_3 + 0x_4 \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \\ x_1, x_2, x_3, x_4 &\geq 0, \end{aligned}$$

and solve the same by the simplex method to get the following tableaus

		x_1	x_2	x_3	x_4
z	0	−5	−2	0	0
x_3	9	2	2	1	0
x_4	11	[3]	1	0	1

		x_1	x_2	x_3	x_4
z	18.33	0	−0.33	0	1.67
x_3	1.67	0	[1.33]	1	−0.67
x_1	3.67	1	0.33	0	0.33

		x_1	x_2	x_3	x_4
z	18.75	0	0	0.25	1.5
x_2	1.25	0	1	0.75	−0.5
x_1	3.25	1	0	−0.25	0.5

Here we have made a minor change in the representation of the simplex tableau. As we have agreed to identify the objective function row by $i = 0$, we have taken the last row (the row having entries as $z(x_B)$ and $(z_j - c_j)$ etc.) of the usual simplex tableau at the top.

In the above tableau we observe that all $(z_j - c_j) \geq 0$ and so an optimal solution of $(LP)_1$ has been obtained. Now we check if this optimal solution meets the integer requirements or not. We note that neither x_1 nor x_2 (and hence z) meet the integer requirements. Therefore we have to obtain the Gomory's cut constraint by choosing a value of i for which f_{i0} is positive most. We observe that $f_{00} = 0.75$, $f_{10} = 0.25$, $f_{20} = 0.25$ and therefore we derive the cut constraint through the objective function row. Further, the nonbasic variables are x_3 and x_4 with f_{03} = fractional part of (0.25) = 0.25 and f_{04} = fractional part of (1.5) = 0.5. Therefore the desired Gomory's cut constraint is

$$f_{00} - f_{03}x_3 - f_{04}x_4 \leq 0,$$

i.e.

$$0.75 - 0.25x_3 - 0.5x_4 \leq 0,$$

i.e.

$$0.75 - 0.25x_3 - 0.5x_4 + s_1 = 0,$$

i.e.

$$-0.75 = s_1 - 0.25x_3 - 0.5x_4. \tag{6.17}$$

We next append this cut constraint to $(LP)_1$ to get $(LP)_2$ as

$$\text{Max} \qquad z = 5x_1 + 2x_2$$

subject to

$$2x_1 + 2x_2 + x_3 = 9$$
$$3x_1 + x_2 + x_4 = 11$$
$$-0.25x_3 - 0.5x_4 + s_1 = -0.75,$$
$$x_1, x_2, x_3, x_4, s_1 \geq 0.$$

Second iteration

We solve problem $(LP)_2$ by the dual simplex method, by taking the constraint (6.17) directly (as it is already in the canonical form and so will be subsequent cut constraints as well) in the last tableau of $(LP)_1$ to get the following dual simplex tableaus

		x_1	x_2	x_3	x_4	s_1
z	18.75	0	0	0.25	1.5	0
x_2	1.25	0	1	0.75	−0.5	0
x_1	3.25	1	0	−0.25	0.5	0
s_1	−0.75	0	0	**−0.25**	−0.5	1

		x_1	x_2	x_3	x_4	s_1
z	18	0	0	0	1	1
x_2	−1	0	1	0	**−2**	3
x_1	4	1	0	0	1	−1
x_3	3	0	0	1	2	4

		x_1	x_2	x_3	x_4	s_1
z	17.5	0	0.5	0	0	2.5
x_4	0.5	0	−0.5	0	1	−1.5
x_1	3.5	1	0.5	0	0	0.5
x_3	2	0	1	1	0	−1

The above tableau gives the optimal solution of $(LP)_2$ which still does not meet the integer requirements. Therefore we have to derive the next cut constraint, by choosing an appropriate value of i. As $f_{00} = 0.5 = f_{10} = f_{20}$, we can generate the cut constraint through z or x_4 or x_1. Suppose we again choose $i = 0$ and derive the cut constraint through the objective function row to get

$$0.5 - 0.5x_2 - 0.5s_1 \leq 0,$$

i.e.

$$0.5 - 0.5x_2 - 0.5s_1 + s_2 = 0,$$

i.e.

$$-0.5 = s_2 - 0.5x_2 - 0.5s_1, \tag{6.18}$$

which we append to $(LP)_2$ to get the next LPP, namely $(LP)_3$.

Third iteration

The new LPP, $(LP)_3$, is solved by the dual simplex method by taking the cut constraint (6.18) directly into the last tableau of $(LP)_2$. This gives the following dual simplex tableaus

		x_1	x_2	x_3	x_4	s_1	s_2
z	17.5	0	0.5	0	0	2.5	0
x_4	0.5	0	−0.5	0	1	−1.5	0
x_1	3.5	1	0.5	0	0	0.5	0
x_3	2	0	1	1	0	−1	0
s_2	−0.5	0	−0.5	0	0	−0.5	1

		x_1	x_2	x_3	x_4	s_1	s_2
z	17	0	0	0	0	2	1
x_4	1	0	0	0	1	−1	−1
x_1	3	1	0	0	0	0	1
x_3	1	0	0	1	0	−2	2
x_2	1	0	1	0	0	1	−2

Now the optimal solution of $(LP)_3$, as obtained above, meets all the integer requirements so we stop and declare $(x_1^\star = 3,\ x_2^\star = 1)$ as an optimal solution of the given AILP (6.15) with the objective function value $z^\star = 17$.

As our original AILP involves only two variables x_1 and x_2 we can plot the feasible region as well as all cut constraints graphically and visualize that the kth cut constraint certainly deletes a part of the feasible region of problem $(LP)_k$ but does not delete any integer feasible point of the original AILP (6.15).

In Fig 6.3, the polytope ODAF is the feasible region of the associated LPP, $(LP)_1$ and its optimal solution is the corner point $A : (3.25, 1.25)$ as obtained by the simplex method. Since this point does not meet the integer requirements, we derived the first Gomory's cut constraint given at (6.17), namely $0.25x_3 + 0.5x_4 \geq 0.75$. The first thing we wish to do is to show this constraint graphically in Fig 6.3. Some readers may think that probably it is not possible because in Fig 6.3, the coordinate axis are x_1 and x_2 whereas the cut constraint is in the variables x_3 and x_4. But there is no such difficulty because it is known that $x_3 = 9 - 2x_1 - 2x_2$ and $x_4 = 11 - 3x_1 - x_2$. On substituting for x_3 and x_4 in terms of x_1 and x_2, we observe that the given cut constraint $0.25x_3 + 0.5x_4 \geq 0.75$ is in fact $2x_1 + x_2 \leq 7$ as depicted in Fig 6.3. This constraint deletes the region BDAE

from the original polytope ODAF and the remaining region, i.e. the polytope OBEF becomes the feasible region of problem $(LP)_2$. It can be verified that the deleted region BDAE does not have any point with integer coordinates.

In the second iteration, problem $(LP)_2$ is solved by the dual simplex method to get its optimal solution as $(x_1 = 3.5,\ x_2 = 0)$ which is the corner point B in Fig 6.3. As this solution again does not meet the integer requirements we obtained the second cut constraint given at (6.18), i.e.$0.5x_2 + 0.5s_1 \geq 0.5$. Now to plot this cut constraint we have to express s_1 in terms of x_1 and x_2. From (6.17) we have $s_1 = -0.75 + 0.25x_3 + 0.5x_4$. This together with $x_3 = 9 - 2x_1 - 2x_2,\ x_4 = 11 - 3x_1 - x_2$ gives the second cut constraint as $x_1 \leq 3$, which is again shown graphically in Fig 6.3. This cut constraint deletes the region GBC from the polytope OBEF to give the remaining polytope OGCEF as the feasible region of problem $(LP)_3$. Here again we may verify that the deleted region GBC does not have any point with integer coordinates.

In the third iteration, we solve problem $(LP)_3$ by the dual simplex method to get the corner point $C : (3, 1)$. As this point meets the integer requirements of the given AILP, it is optimal for the same and therefore we stop.

Remark 6.3.3. By taking L.C.M., the cut constraint (6.17) can also be rewritten as $-3 = 4\ s_1 - x_3 - 2x_4$. If we further agree to denote $4s_1$ as new s_1 then this equation becomes $-3 = s_1 - x_3 - 2x_4$, which on substitution for x_3 and x_4 in terms of x_1 and x_2 again gives $2x_1 + x_2 \geq 7$. Therefore geometrically it does not matter whether we take L.C.M. or not. Algebraically also it does not matter whether the appended cut constraint is taken as given at (6.17) or in its equivalent form $-3 = s_1 - x_3 - 2x_4$. Also after getting an optimal solution of $(LP)_k$, for stopping we do not have to bother whether s_1, s_2 etc. are integers or not. This is because we are solving the AILP (6.5) where only the variables x are constrained to be integers, and therefore for stopping we have to check that all components of x are integers in the optimal solution of $(LP)_k$. However it is obvious that as the problem is all integer, the variables s_1, s_2 etc. are automatically constrained to be integers if the cut constraints are appended after taking L.C.M. in (6.13) or equivalently in (6.14) and renaming the new s_i again as s_i.

6.4 Gomory's Cutting Plane Method for Mixed Integer Linear Programming

In this section we extended the ideas of Section 6.3 to the mixed integer case so as to obtain the Gomory's cutting plane method for the mixed integer linear programming (MILP) problem.

Let the given MILP have the following form

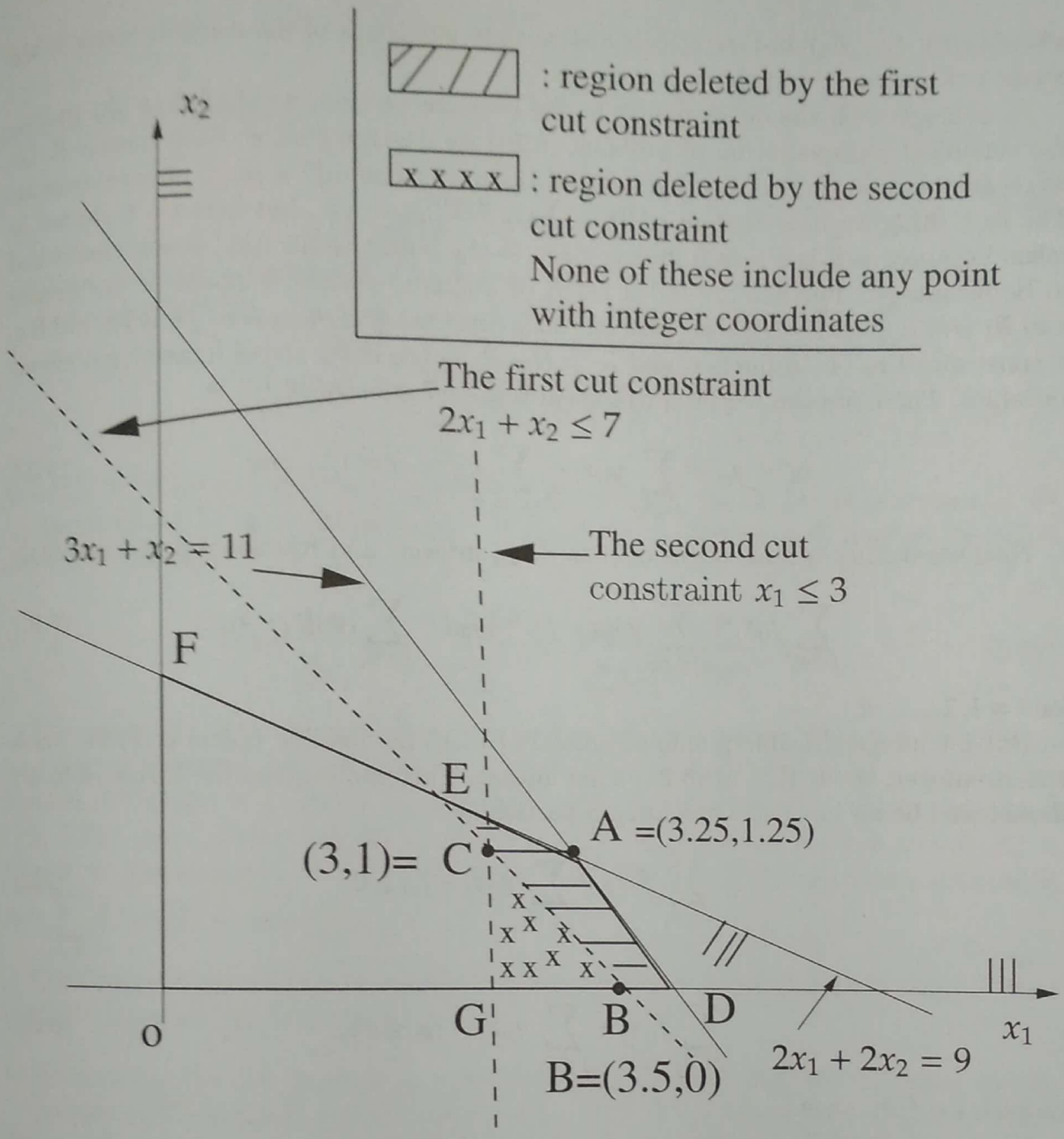


Fig. 6.3.

$$\begin{aligned} \text{Max} \quad & z = c_1^T x + c_2^T v \\ \text{subject to} \quad & \\ & A_1 x + A_2 v = b \\ & x \geq 0, v \geq 0 \\ & x \text{ integer}, \end{aligned} \tag{6.19}$$

where (x, v), (A_1, A_2) and (c_1, c_2) are appropriate partitions of the decision vector x, the coefficient matrix A and the cost vector c.

In analogy with the development in the previous section, we shall first like to have the canonical representation of problem (6.19) for a given feasible basis matrix B. Let us recall equation (6.9) of Section 6.3 and then try to modify it in an obvious manner. The first thing we note that in (6.9) we have $i = 0, 1, \ldots, m$, but here $i = 0$ cannot be taken because, as it is a mixed integer case, so the objective function is not constrained to be an integer. Further the index set R of nonbasic variables should be partitioned into R_1 and R_2, i.e. $R = R_1 \cup R_2$, such that R_1 consists of those indices $j \in R$ for which x_j is constrained to be an integer, and $R_2 = R \sim R_1$ is the index set of remaining nonbasic variables. Therefore the required canonical representation of (6.19) is

$$x_{B_i} = y_{i0} - \sum_{j \in R_1} y_{ij} x_j - \sum_{j \in R_2} y_{ij} v_j, \ (i = 1, \ldots, m). \tag{6.20}$$

Now separating various terms of (6.20) into integers and fractional parts, we obtain

$$\sum_{j \in R_1} f_{ij} x_j + \sum_{j \in R_2} y_{ij} v_j - f_{i0} = [y_{i0}] - \sum_{j \in R_1} [y_{ij}] x_j - x_{B_i} \tag{6.21}$$

for $i = 1, 2, \ldots, m$.

But for integer feasible points of problem (6.19), i.e. feasible points (x, v) for which x is an integer, the R.H.S. of (6.21) is an integer. This implies that the L.H.S. of (6.21) should also be an integer. Therefore, in particular

$$\sum_{j \in R_1} f_{ij} x_j + \sum_{j \in R_2} y_{ij} v_j - f_{i0} \geq 0 \tag{6.22}$$

or

$$\sum_{j \in R_1} f_{ij} x_j + \sum_{j \in R_2} y_{ij} v_j - f_{i0} \leq -1, \tag{6.23}$$

for each $i = 1, 2, \ldots, m$.

We now define following two sets for each $i = 1, \ldots, m$,

$$R_2^+ = \{j \in R_2 : y_{ij} > 0\}$$

and

$$R_2^- = \{j \in R_2 : y_{ij} < 0\}.$$

Then if (6.22) holds, so does

$$\sum_{j \in R_1} f_{ij} x_j + \sum_{j \in R_2^+} y_{ij} v_j \geq f_{i0}, \tag{6.24}$$

and if (6.23) holds so does

$$\sum_{j \in R_2^-} y_{ij} v_j \leq (-1 + f_{i0}), \tag{6.25}$$

because $x_j \geq 0$ and $f_{ij} \geq 0$ for all $j \in R_1$.

If we multiply (6.25) by $(f_{i0}/(-1+f_{i0}))$ (which is less than zero as $0 < f_{i0} < 1$), we get

$$-\sum_{j \in R_2^-} \frac{f_{i0} y_{ij} v_j}{(1 - f_{i0})} \geq f_{i0}. \tag{6.26}$$

As L.H.S. of (6.24) and (6.26) are both non negative and exactly one of these must hold (i.e. either (6.24) holds or (6.26) holds - both obviously cannot hold because of (6.22) and (6.23)) we have

$$f_{i0} - \sum_{j \in R_1} f_{ij} x_j - \sum_{j \in R_2^+} y_{ij} v_j + \sum_{j \in R_2^-} \frac{f_{i0} y_{ij} v_j}{(1 - f_{i0})} \leq 0. \tag{6.27}$$

The constraint (6.27) is called the *Gomory cut constraint for MILP*. It is certainly a cut constraint because it is satisfied by every feasible point (x, v), x integer, of the given problem (6.19) and it is not satisfied by the current optimal solution of the associated LPP, $(LP)_1$. Also, if the problem is all integer then $R_1 = R$ and $R_2 = \emptyset$ and so $R_2^+ = R_2^- = \emptyset$. In this situation the cut constraint (6.27) reduces to the usual cut constraint $f_{i0} - \sum_{j \in R} f_{ij} x_j \leq 0$, as expected.

The Gomory's cutting plane method for solving MILP remains exactly same as that of AILP except for the following

(i) for deriving the cut constraint, equation (6.27) is used and not equation (6.13), because (6.27) is the Gomory's cut constraint for MILP, whereas (6.13) is for AILP.

(ii) for generating the cut constraint (6.27), $i = 0$ is never chosen, because $i = 0$ represents the objective function which is not guaranteed to be integer for the mixed integer case even if the components of the cost vector c are integers.

(iii) excluding the value $i = 0$, we choose that i for which the variable x_{B_i} is constrained to be integer but is currently having the fractional value. Amongst these, we choose the one for which the fractional part f_{i0} is positive most.

(iv) once an optimal solution of $(LP)_k$ has been obtained, we stop if those variables which are constrained to be integers are having integer value, and we do not bother about the integrability of other variables.

Example 6.4.1. Use Gomory's cutting plane method to solve the following mixed integer LPP

$$\begin{aligned} \text{Max} \quad & z = 2x_1 + x_2 \\ \text{subject to} \quad & \\ & x_1 + x_2 \leq 5 \\ & 6x_1 + 2x_2 \leq 21 \\ & x_1, x_2 \geq 0 \\ & x_1 \text{ integer.} \end{aligned}$$

Solution The given problem is

$$\begin{aligned} \text{Max} \quad & z = 2x_1 + x_2 + 0x_3 + 0x_4 \\ \text{subject to} \quad & \\ & x_1 + x_2 + x_3 = 5 \\ & 6x_1 + 2x_2 + x_4 = 21 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1 \text{ integer .} \end{aligned}$$

We shall solve the above problem by using the Gomory's cutting plane method for MILP. For this we consider the associated LPP, $(LP)_1$ and solve the same by the simplex method to get the following optimal simplex tableau

		x_1	x_2	x_3	x_4
z	31/4	0	0	1/2	1/4
x_1	11/4	1	0	−1/2	1/4
x_2	9/4	0	1	3/2	−1/4

Now the optimal solution of $(LP)_1$ does not meet the integer requirements (as x_1 is constrained to be integer but is having a fractional value) and therefore it is not optimal to the given integer programming problem. So we have to derive the cut constraints (6.27) by choosing an appropriate value of i. Here only $i = 1$ (i.e. through the variable x_1) is the only possibility as only x_1 is constrained to be integer. Therefore we identify R_1, R_2^+, R_2^- etc. to get

$$\begin{aligned} & R_1 = \emptyset \\ & R_2 = R \sim R_1 = R = \{3, 4\} \\ & R_2^+ = \{4\},\ R_2^- = \{3\} \\ & y_{14} = 1/4,\ y_{13} = -1/2,\ f_{10} = 3/4. \end{aligned}$$

In the above, $R_1 = \emptyset$ because by definition, R_1 is the index set of those nonbasic variables which are constrained to be integers, and from the tableau, the nonbasic variables are x_3 and x_4, and none of these are constrained to be integers. Next $R_2 = R$ because

$R_2 = R \sim R_1 = R \sim \emptyset = R$. Also, by definition, R_2^+ is the index set of those nonbasic variables which are in R_2 and for which $y_{ij} > 0$. From the tableau, only such nonbasic variable is x_4 and so R_2^+ is the index set of x_4 i.e. $R_2^+ = \{4\}$. In a similar manner $R_2^- = \{3\}$. Plugging these values in (6.27) we get the cut constraint as

$$\frac{1}{4}x_4 - \frac{3/4}{(1-3/4)}(-1/2)x_3 \geq \frac{3}{4}$$

i.e.

$$-\frac{3}{4} = s_1 - \frac{1}{4}x_4 - \frac{3}{2}x_3.$$

We next append the above cut constraint in $(LP)_1$ to get the new problem $(LP)_2$ which we solve by employing the dual simplex method. This gives the following tableaus

		x_1	x_2	x_3	x_4	s_1
z	31/4	0	0	1/2	1/4	0
x_1	11/4	1	0	−1/2	1/4	0
x_2	9/4	0	1	3/2	−1/4	0
s_1	−3/4	0	0	**−3/2**	−1/4	1

		x_1	x_2	x_3	x_4	s_1
z	30/4	0	0	0	1/6	1/3
x_1	3	1	0	0	1/3	−1/3
x_2	3/2	0	1	0	−1/2	1
x_3	1/2	0	0	1	1/6	−2/3

So the optimal solution of $(LP)_2$ is $(x_1 = 3, x_2 = 3/2, x_4 = 3/2, x_3 = 1/2)$, which meets the integer requirements as x_1 takes the integer value, namely $x_1 = 3$. So we stop and get an optimal solution of the given MILP as $(x_1^\star = 3, x_2^\star = 3/2)$ with the optimal value $z^\star = 30/4$.

Now, similar to Example 6.3.2, here again we can have a graphical visualization of the feasible region and cut constraints etc. to verify that the cut constraint $(1/4)x_4+(3/2)x_3 \geq 3/4$ certainly deletes a part of the feasible region $(LP)_1$ but does not delete any integer feasible point, i.e. any feasible point (x_1, x_2) for which x_1 is an integer. We refer to Fig 6.4 in this regard.

In this figure, the polytope OABC is the feasible region of the associated LPP, $(LP)_1$ and the bold lines consists of those points in this region for which the first coordinate x_1 is an integer, i.e. these lines represent the feasible set of the given MILP. After substituting for x_3 and x_5 in terms of x_1 and x_2, we get the cut constraint as $3x_1 + 2x_2 \leq 12$ which is shown by the dotted line in Fig 6.4. Therefore the feasible region for the new problem $(LP)_2$ comes out to be the polytope OADEC and the optimal solution of $(LP)_2$ comes

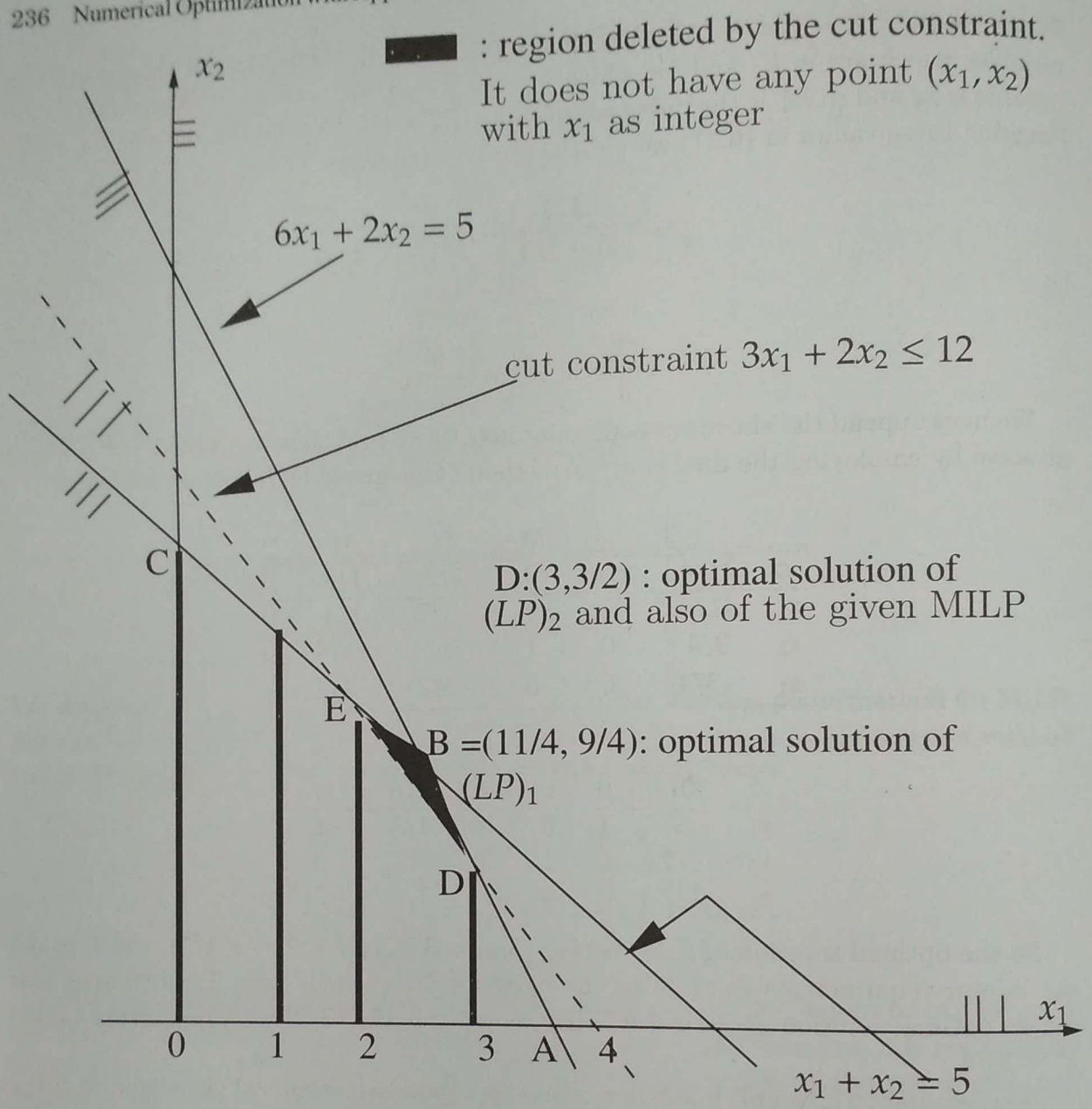


Fig. 6.4.

out to be $(x_1 = 3, x_2 = 3/2)$ attained at the corner point D. Further, the deleted region DBE does not include any point (x_1, x_2) for which x_1 is an integer.

6.5 Branch and Bound Method

Branch and bound is an efficient enumerative technique for examining all possible integer feasible points and this can be used to solve both, all as well as mixed, types of integer programming problems. Let the given ILP be

$$\begin{aligned} &\text{Max} \quad z = c^T x \\ &\text{subject to} \\ &\qquad Ax = b \\ &\qquad x \geq 0 \\ &x_j \text{ integer for } j \in J_1 \subset J = \{1, \ldots, n\}. \end{aligned} \tag{6.28}$$

As the name suggests, each iteration consists of *branching* and *bounding*. The branching is done from the current *node*, a node being identified as a LPP, where the initial node is the associated LPP of the given problem (6.28). Certain appropriate bounds are calculated so as to decide if further branching is needed or not. Though it is an enumerative technique, it is efficient in the sense that the exhaustive enumeration is seldom needed. Most of the times it is only partial enumeration as that node which cannot further improve the current available solution is discarded and therefore the corresponding region is no more required for enumeration. The following are the details of the Branch And Bound method.

Step 1 Solve the associated LPP, called $(LP)_1$ and let z_1 be its optimal value. If the optimal solution of $(LP)_1$ has integer components for $j \in J_1$, i.e. it meets the integer requirements of the given ILP, stop; otherwise go to Step 2. Thus in Step 1, either we stop and get an optimal solution of the given ILP or we have an *upper bound* z_1 for the optimal objective function value of the ILP.

Step 2 Select a variable x_j which is constrained to be integer but is currently having a fractional value β_j in the optimal $(LP)_1$ solution, and construct the following two LP's

$\underline{(LP)_2}$	$\underline{(LP)_3}$
Max $z = c^T x$	Max $z = c^T x$
subject to	subject to
$Ax = b$	$Ax = b$
$x \geq 0$	$x \geq 0$
$x_j \leq [\beta_j]$.	$x_j \geq < \beta_j >$.

Here $[\beta_j]$ is the *greatest integer less than or equal to* β_j and $< \beta_j >$ is the *smallest integer more than or equal to* β_j. Obviously $[\beta_j]$ is the nearest integer in the left of β_j and $< \beta_j >$ is the nearest integer in the right of β_j. For example if $\beta_j = 1.6$ then $[\beta_j] = 1$ and $< \beta_j >= 2$. Sometimes $[\beta_j]$ and $< \beta_j >$ are also called, respectively, the *floor* and *ceiling* of β_j.

If we decide to identify problem $(LP)_k$ by the kth node, then the above process is to branch from node 1 via the variable x_j to get two new nodes, node 2 and node 3. The new restrictions $x_j \leq [\beta_j]$ and $x_j \geq < \beta_j >$, are mutually exclusive so that $(LP)_2$ and $(LP)_3$ are dealt with separate LP's. The optimal solution of the given ILP lies either in $(LP)_2$ or $(LP)_3$, i.e. in one of the two branches as shown in Fig. 6.5.

Step 3 Obtain an optimal solution of $(LP)_2$ and $(LP)_3$. Clearly $(LP)_2$ and $(LP)_3$ should be solved by the dual simplex method as both of these LP's differ from $(LP)_1$ with regard

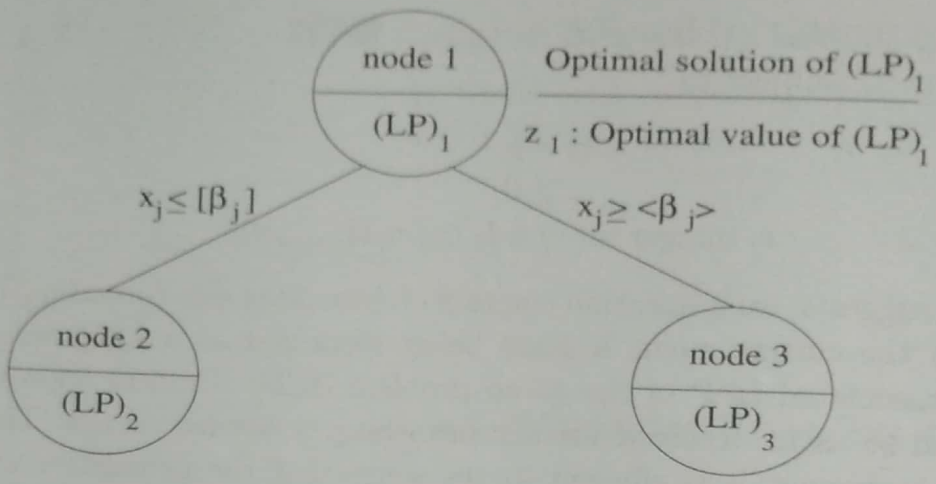


Fig. 6.5.

to an additional constraint only. Suppose that optimal solutions of $(LP)_2$ and $(LP)_3$ still do not meet the integer requirements of the given ILP.

Step 4 Select either $(LP)_2$ or $(LP)_3$ for further branching. Branch from these by adding a new constraint corresponding to a variable which is constrained to be integer but is having a fractional value.

Step 5 This process of branching and solving a sequence of LP's is continued until a solution is obtained for one of the LP's which meets the integer requirements, namely x_j integer for $j \in J_1$. Let this LPP be $(LP)_k$ and z_k denote its optimal value. Then z_k becomes a *lower bound* for the optimal objective function value of the given ILP.

At this point, we can eliminate from consideration all those LP's (nodes) which have the objective function value z less than this lower bound, namely z_k. We say that all such nodes have been *fathomed* because it is not possible to find a better solution (lower bound) from these LP's than what we have now. This is because, branching from a LPP means adding one more constraint and hence further reducing its feasible region.

Step 6 Continue selecting a node (i.e. LP) and branching till all the nodes have been fathomed. The fathomed node with the largest value of z will give the optimal solution of the given ILP.

Example 6.5.1. Use branch and bound method to solve the following ILP

$$\text{Max} \quad z = 7x_1 + 9x_2$$

subject to

$$\begin{aligned} -x_1 + 3x_2 &\le 6 \\ 7x_1 + x_2 &\le 35 \\ x_1 &\le 7 \\ x_2 &\le 7 \\ x_1, x_2 &\ge 0 \\ x_1 \text{ and } x_2 &\text{ integer.} \end{aligned}$$

Solution The first step is to solve the associated LPP, $(LP)_1$. The optimal solution of $(LP)_1$ is $(x_1 = 9/2, x_2 = 7/2)$ and the optimal value is $z_1 = 63$. As the solution of $(LP)_1$ does not meet the integer requirements, it is not optimal for the given ILP. However, $z_1 = 63$ is an upper bound for the optimal objective function value of the given ILP. problem $(LP)_1$ is depicted as node 1 in Fig 6.5.

The next step of the branch and bound method is to branch from node 1 via a variable x_j which is constrained to be integer but is currently having a fractional value. In our example, x_1 and x_2 both are constrained to be integers and both are having a fractional value so we may branch from node 1 via x_1 or x_2. In practice, we choose that j for which the fractional part of such x_j is positive most, with the hope that this way we may get a *deeper* cut. Here both x_1 and x_2 have equal fractional part so we can branch either from x_1 or x_2. If we decide to branch from x_1, then for $\beta_1 = 9/2$, $[\beta_1] = 4$ and $< \beta_1 >= 5$ and therefore we get two new LP's, $(LP)_2$ and $(LP)_3$, as

$(LP)_2$	$(LP)_3$
Same as $(LP)_1$ but with one additional constraint $x_1 \le 4$.	Same as $(LP)_1$ but with one additional constraint $x_1 \ge 5$.

These are identified as node 2 and node 3. Solving $(LP)_2$ we get $(x_1 = 4, x_2 = 10/3)$ and $z_2 = 58$. Similarly solving $(LP)_3$ we get $(x_1 = 5, x_2 = 0)$ and $z_3 = 35$. As the solution of $(LP)_3$ meets the integer requirements, the value $z_3 = 35$ becomes a lower bound for the optimal objective function value of the given ILP.

At this stage we check the objective function value of other LP's. In our case, there is only one node to be checked, i.e. node 2. As the objective function value of $(LP)_2$ is more than this lower bound $z_3 = 35$, this node is not fathomed and it is considered for further branching. Had the value of $(LP)_2$ been less than the lower bound $z_3 = 35$, it would have been a fathomed node and $(x_1 = 5, x_2 = 0)$ would have been taken as an optimal solution of the given ILP.

Branching from node 2, via x_2 we get two LP's, namely $(LP)_4$ and $(LP)_5$. problem $(LP)_5$ is infeasible and an optimal solution of $(LP)_4$ is $(x_1 = 4, x_2 = 3)$ with the objective function value $z_4 = 55$.

Now all nodes have been fathomed and so we have to choose the fathomed node with the maximum objective function value. This gives $(x_1^\star = 4, x_2^\star = 3)$ as an optimal solution of the given ILP and the objective function value $z^\star$ as $z^\star = 55$.

We depict the above details in Fig 6.6.

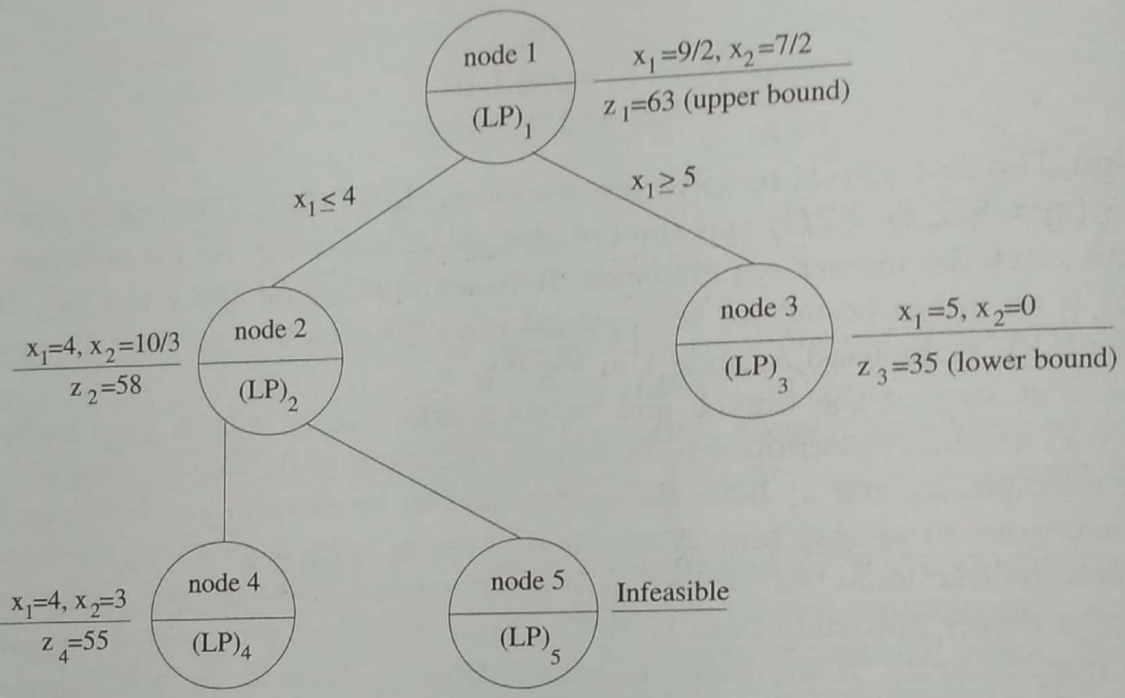


Fig. 6.6.

Example 6.5.2. Use branch and bound method to solve the following integer LPP

$$\text{Max} \quad z = 3x_1 + 4x_2$$

subject to

$$7x_1 + 16x_2 \leq 52$$
$$3x_1 - 2x_2 \leq 9$$
$$x_1, x_2 \geq 0$$
$$x_1 \text{ and } x_2 \text{ integer.}$$

Solution Following BBM we obtain Fig. 6.7

Now all the nodes have been fathomed. Therefore an optimal solution of the given ILP is $(x_1^\star = 2, x_2^\star = 2)$ and the optimal value is $z^\star = 14$. If we plot the feasible region of $(LP)_1$, $(LP)_2$ and $(LP)_3$ then we can visualize that the optimal solution of the ILP is either in $(LP)_1$ or $(LP)_2$, i.e. either in the branch $x_2 \leq 1$ or in the branch $x_2 \geq 2$. Fig 6.8 illustrates this point.

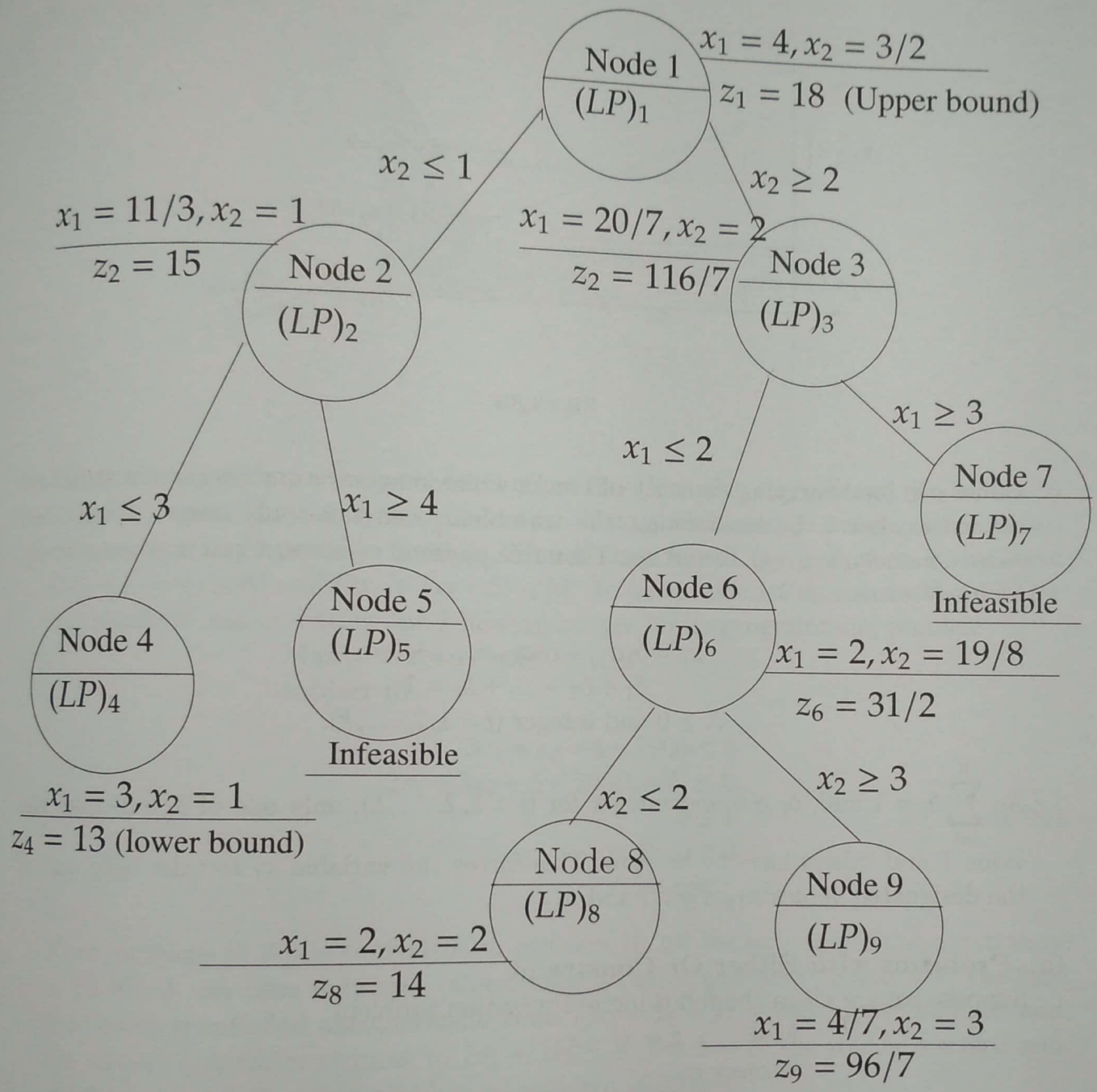


Fig. 6.7.

6.6 Some Other Related Problems

In this section we present a few other interesting optimization problems which as such do not fit into the model of an AILP or MILP but, can be transformed in a manner so that the methods discussed in earlier sections can be applied.

(a) **Discrete Value Linear Programming**

In this class of problems, a variable (say x_j) can take only one of the designated several

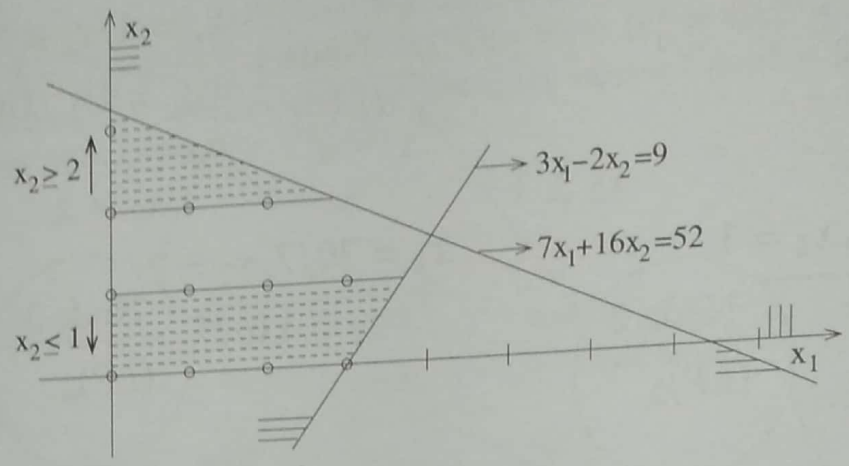


Fig. 6.8.

values e.g. load carrying capacity of trucks which may vary among certain standard capacities. Let it be known that the variable x_j can take only one of k specified values, namely, $\alpha_{1j}, \alpha_{2j}, \ldots$ and α_{kj}. Then this physical constraint can mathematically be modeled as

$$\begin{aligned} x_j &= \delta_1\alpha_{1j} + \delta_2\alpha_{2j} + \ldots + \delta_k\alpha_{kj} \\ \delta_1 &+ \delta_2 + \ldots + \delta_k = 1 \\ \delta_i &\geq 0 \text{ and integer } (i = 1, 2, \ldots, k). \end{aligned}$$

As $\sum_{i=1}^{k} \delta_i = 1$ and, $\delta_i \geq 0$ and integer for $(i = 1, 2, \ldots, k)$, only one of δ_i will take the value 1 and others have to be zero. This forces the variable x_j to take only one of the designated values $\alpha_{1j}, \alpha_{2j}, \ldots$ and α_{kj}.

(b) **Problems with Either Or Constraint**

Suppose we are given the following optimization problem

$$\begin{aligned} &\text{Max} \quad z = x_1 + x_2 \\ &\text{subject to} \\ &\qquad \begin{cases} 3x_1 + x_2 \leq 4 \\ \quad or \\ 2x_1 + 3x_2 \leq 5 \end{cases} \qquad (6.29) \\ &\qquad x_1, x_2 \geq 0. \end{aligned}$$

Let us plot the feasible region S of the above problem.

From Fig 6.9, we note that S is not a convex set and hance problem (6.29) cannot be a linear programming problem. For some readers, it should have been obvious because by definition all constraints in a LPP are in 'and' form, i.e. they are taken together.

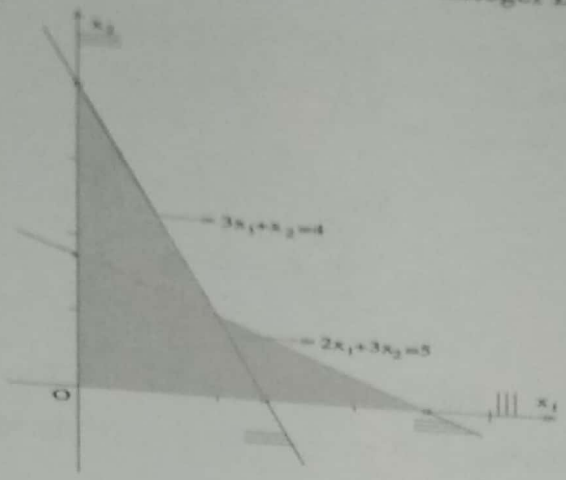


Fig. 6.9.

We now assume that an upper bound of the functions $(3x_1+x_2-4)$ and $(2x_1+3x_2-5)$ over the region $(x_1 \geq 0, x_2 \geq 0)$ is known. If no such bound is known then we can take a very large positive number M such that for all $(x_1 \geq 0, x_2 \geq 0)$ we have $(3x_1 + x_2 - 4) \leq M$ and $(2x_1 + 3x_2 - 5) \leq M$. To be specific let us choose $M = 100$ for our example and construct the following integer linear programming problem

Max $z = x_1 + x_2$

subject to

$$\begin{aligned} 3x_1 + x_2 - 4 - 100\delta &\leq 0 \\ 2x_1 + 3x_2 - 5 - 100(1-\delta) &\leq 0 \\ \delta &\leq 1 \\ x_1, x_2, \delta &\geq 0 \\ \delta &\text{ integer .} \end{aligned} \tag{6.30}$$

Then problem (6.30) is equivalent to problem (6.29) because as per the constraints of (6.30), δ can take only two values, namely 'zero' and 'one'. For $\delta = 1$, the first constraint is satisfied automatically since $(3x_1 + x_2 - 4) \leq 100$ (upper bound), and the second constraint reduces to $2x_1 + 3x_2 \leq 5$. For $\delta = 0$, the role is reversed and therefore we get the same problem as problem (6.29).

In this context it is obvious that if the constraints are given in '$\geq$' form rather than upper bound, we need to lower bound of the constraint functions because if M is an upper bound of $g(x)$, then $-M$ is a lower bound of $-g(x)$ over the same region.

(c) A Specified Number 'p' out of 'm' Given Constraints

Let the given constraints be $g_i(x) \leq 0$, $(i = 1, 2, \ldots, m)$ and we wish that p (any p and not the specified p constraints) of these m constraints should hold. Let M_i be an upper bound for the constraint function g_i, i.e. $g_i(x) \leq M_i$ for $x \in \mathbf{R}^n$. Then the set of constraints

$$
\begin{gathered}
g_1(x) - M_1\delta_1 \leq 0 \\
g_2(x) - M_2\delta_2 \leq 0 \\
\vdots \\
g_m(x) - M_m\delta_m \leq 0 \\
\sum_{i=1}^{m} \delta_i = m - p \\
0 \leq \delta_i \leq 1 \\
\text{and } \delta_i \text{ integer}, (i = 1, 2, \ldots, m).
\end{gathered}
$$

model the physical situation that p out of m given constraints hold.

6.7 Zero-One Implicit Enumeration Algorithm

In the last section, we have seen few examples of $0-1$ ILP's. Though these problems could also be solved by any general ILP methods, e.g. cutting plane or branch and bound methods, we have certain special algorithms exclusively for $0-1$ problem. One such algorithm is called the *Balas' additive algorithm* and that we discuss in this section.

In principle, the additive algorithm is a special case of the general branch and bound strategy, but it is different in implementation. It does not require solving LP's and the main computational step needs only some simple additions and subtractions to use the heuristics.

To apply the additive algorithm, we assume that the given $0-1$ ILP satisfies the following requirements

(i) The objective function is given in the minimization form, i.e. min $c^T x$, with all coefficients $c_j \geq 0$.

(ii) All the constraints are given in '$\leq$' form. This necessarily means that some b_i may be less than or equal to zero. These constraints are then converted to equations by introducing slack variables s_i, such that $s_i \geq 0$ for all i.

Once the given $0-1$ ILP is re-casted in the form so as to meet the above requirements, we say that the problem is given in the *standard* $0-1$ *ILP form*. We use additive algorithm to solve only those $0-1$ ILP's which are in the standard $0-1$ ILP form.

We now illustrate that any $0-1$ ILP can always be transformed to the standard $0-1$ ILP form.

Example 6.7.1. Transform the following to the standard $0-1$ ILP form

$$
\begin{aligned}
&\text{Max} \quad z = 3x_1 - 5x_2 \\
&\text{subject to}
\end{aligned}
$$

$$
\begin{gathered}
x_1 + x_2 = 5 \\
4x_1 + 6x_2 \geq 4 \\
x_1, x_2 \text{ both are 0 or 1.}
\end{gathered} \tag{6.31}
$$

Solution We observe that (*i*) $Max\ 3x_1 - 5x_2$ is equivalent to $-Min\ (-3x_1 + 5x_2)$ (*ii*) $x_1 + x_2 = 5$ is equivalent to $x_1 + x_2 \leq 5$ and $-x_1 - x_2 \leq -5$ and (*iii*) $4x_1 + 6x_2 \geq 4$ is equivalent to $-4x_1 - 6x_2 \leq -4$. Therefore problem (6.31) is equivalent to

$$\begin{aligned} &\text{Min} && w = -3x_1 + 5x_2 \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} x_1 + x_2 + s_1 &= 5 \\ -x_1 - x_2 + s_2 &= -5 \\ -4x_1 - 6x_2 + s_3 &= -4 \\ x_1, x_2 &\text{ both are } 0 \text{ or } 1 \\ s_1 \geq 0, s_2 \geq 0, s_3 &\geq 0. \end{aligned} \tag{6.32}$$

The only problem now is that the coefficient of x_1 in the objective function is -3 which is less than zero. But if we substitute $x_1 = (1 - x_1')$ in the objective function and adjust the R.H.S of the constraints of (6.32) accordingly we get the problem

$$\begin{aligned} &\text{Min} && w' = 3x_1' + 5x_2 \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} -x_1' + x_2 + s_1 &= 4 \\ x_1' - x_2 + s_2 &= -4 \\ 4x_1' - 6x_2 + s_3 &= 0 \\ x_1', x_2 &\text{ both are } 0 \text{ or } 1 \\ s_1 \geq 0, s_2 \geq 0, s_3 &\geq 0. \end{aligned}$$

where $w = w' + 3$, and minimizing w is same as minimizing w'. This problem is in the standard $0 - 1$ ILP form.

The branching strategy of the additive algorithm is again based on the use of a branching variable x_j. Here the two branches are corresponding to $x_j = 0$ and $x_j = 1$ as x_j is a binary variable. The bounding strategy is very similar to the branch and bound method. Hence an improved integer solution provides an upper bound on the minimum value of the objective function.

In any subproblem, the fathoming can occur in any one of the following three ways

(i) The subproblem cannot lead to a feasible solution.
(ii) The subproblem cannot yield a better upper bound.
(iii) The subproblem leads to a feasible integer solution.

We now illustrate the working of the additive algorithm.

Example 6.7.2. Solve the following $0-1$ ILP by the additive algorithm

$$\text{Max} \quad w = 3y_1 + 2y_2 - 5y_3 - 2y_4 + 3y_5$$

subject to

$$\begin{aligned} y_1 + y_2 + y_3 + 2y_4 + y_5 &\le 4 \\ 7y_1 + 3y_3 - 4y_4 + 3y_5 &\le 8 \\ 11y_1 - 6y_2 + 3y_4 - 3y_5 &\ge 3 \\ y_1, y_2, y_3, y_4, y_5 &\text{ all 0 or 1.} \end{aligned}$$

Solution The standard $0-1$ ILP form of the above problem is

$$\text{Min} \quad z = 3x_1 + 2x_2 + 5x_3 + 2x_4 + 3x_5$$

subject to

$$\begin{aligned} -x_1 - x_2 + x_3 + 2x_4 - x_5 + s_1 &= 1 \\ -7x_1 + 3x_3 - 4x_4 - 3x_5 + s_2 &= -2 \\ 11x_1 - 6x_2 - 3x_4 - 3x_5 + s_3 &= -1 \\ x_1, x_2, x_3, x_4, x_5 &\text{ all } 0 \text{ or } 1 \\ s_1, s_2, s_3 &\text{ all } \ge 0, \end{aligned} \tag{6.33}$$

where $y_1 = 1 - x_1$, $y_2 = 1 - x_2$, $y_3 = x_3$, $y_4 = x_4$, $y_5 = 1 - x_5$ and $w = z - 8$.

Now we explain the method. As in problem (6.33), we seek the minimization of a linear objective function with all coefficients non-negative, a logical starting solution should consists of all-zero binary variables. In this case the slacks will act as basic variables whose values will be b_i's given on the R.H.S. This gives the following tableau

Basic	x_1	x_2	x_3	x_4	x_5	s_1	s_2	s_3	solution
s_1	−1	−1	1	2	−1	1	0	0	1
s_2	−7	0	3	−4	−3	0	1	0	−2
s_3	11	−6	0	−3	−3	0	0	1	−1
Objective coefficients	3	2	5	2	3				

Now given an initial all-zero binary solution, the associated slack solution is $(s_1, s_2, s_3) = (1, -2, -1)$, $z = 0$. In case all the slacks were non-negative, we would have stopped and declared all-zero binary solution as optimal.

However, as some slacks are negative (infeasible) we need to upgrade one or more binary variables to the value of '1' to achieve feasibility or to declare that the problem is infeasible.

In additive algorithm as well, we elevate only *one* of the binary variables from 'zero' value to the value '1'. The chosen variable is called the branching variable and its solution is based on the use of certain special tests as given below.

(i) The branching variable should be capable of reducing the infeasibility of slacks. Looking at the above tableau, we note that x_3 cannot be selected as a branching

variable, because its coefficient in the second and third constraints is non-negative. Therefore setting $x_3 = 1$ can only make s_2 and s_3 more infeasible. For other remaining variables, namely x_1, x_2, x_4 and x_5, there is at least one negative coefficient in the constraint 2 and constraint 3, so they are the possible candidates for the elevation to the value 'one'.

(ii) Amongst the possible candidates for branching (e.g. x_1, x_2, x_4 and x_5 in the given example), we evaluate the *measure of slack infeasibility*. This measure is based on the assumption that a zero value variable x_j will be elevated to the value 'one' and it is defined as

$$I_j = \sum_{\text{all } i} min(0, s_i - a_{ij}),$$

where s_i is the current value of the slack and a_{ij} is the coefficient of the variable x_j in the ith constraint. It can be shown that I_j has an equivalent expression, namely

$$\begin{aligned} I_j = &\text{ sum of the negative slacks resulting from elevating } x_j \text{ to value 'one'} \\ = &\sum_{\text{all } i} (\text{negative } s_i \text{ value given } x_j = 1). \end{aligned} \tag{6.34}$$

In our example when we set $x_1 = 1$ we get $s_1 = 1 - (-1) = 2$, $s_2 = -2 - (-7) = 5$ and $s_3 = -1 - 11 = -12$. Thus $I_1 = -12$. Similarly $I_2 = -2$, $I_4 = -1$ and $I_5 = 0$. We do not compute I_3 as x_3 is excluded as a branching variable. Because I_5 yields the smallest measure of slack infeasibility, x_5 is selected as a branching variable. In other words, we compute the largest of I_j's and then decide the branching variable. So from node 0 ($z = 0$) we branch to two new nodes, node 1 and node 2 via the branching variable x_5, by taking $x_5 = 0$ on one branch and $x_5 = 1$ on the other branch. At node 1, we have slacks $(s_1, s_2, s_3) = (2, 1, 2)$ and $z = 3$. Thus node 1 is fathomed and $z = 3$ is the current upper bound for the optimal objective function value. At node 2, we have slacks $(s_1, s_2, s_3) = (-1, 2, -1)$, $z = 0$, which is infeasible. Now the variables x_1, x_2, x_3 and x_4 are the candidates for the branching variable from node 2. Here we note that though solutions at node 2 and node 0 are identical, node 2 is different because x_5 is no longer a branching candidate. Now for node 2, x_3 is not promising because it does not move s_2 or s_3 toward feasibility. The variables x_1 and x_3 are also not promising because their objective coefficients (3 and 5) will yield a worst objective function value, than the current upper bound ($z = 3$). For the remaining variables x_2 and x_4 we have $I_2 = -2$ and $I_4 = -1$, and so x_4 is the branching variable at node 2. At node 3 (defined by fixing $x_5 = 0$ and $x_4 = 1$) we get $(s_1, s_2, s_3) = (-1, 2, 2)$, $z = 2$ which is infeasible. The candidates for branching are x_1, x_2 and x_3. We can check that elevating any of these to the value one' will worsen the value of z (relative to the current upper bound $z = 3$). Thus all possible candidates for branching are excluded, so node 3 is fathomed.

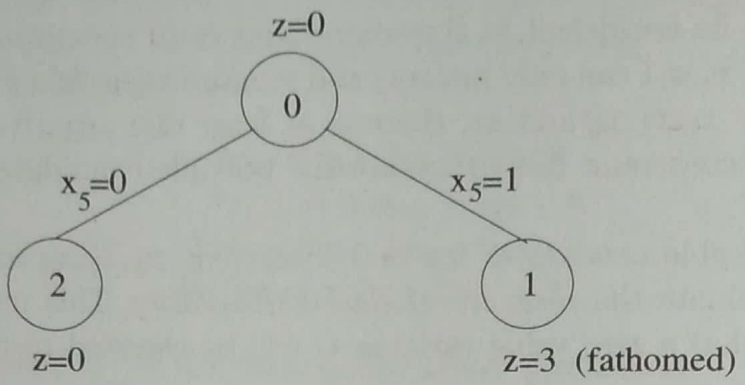


Fig. 6.10.

At node 4, we have $x_5 = x_4 = 0$, which gives $(s_1, s_2, s_3) = (1, -2, -1)$, $z = 0$. The variables x_1 and x_3 are excluded by the upper bound test. In fact, x_3 can also be excluded because it cannot reduce slack infeasibility. The remaining variable x_2 cannot be excluded by any of these two tests so x_2 is chosen as the branching variable. At node 5, we have $(s_1, s_2, s_3) = (2, -2, 5)$, $z = 2$. At this node x_1 and x_3 are branching candidates but they are excluded by the two tests discussed above. So node 5 is fathomed. Also node 6 is fathomed because neither x_1 nor x_3 can produce a better feasible solution. This gives the optimal solution as $x_5 = 1$, other $x_i = 0$ with the optimal value as $z = 3$, given at node 2. Now changing to y_i etc., we get $y_1 = 1$, $y_2 = 1$, $y_3 = 0$, $y_4 = 0$, $y_5 = 0$ with $w = -(z - 8) = -(3 - 8) = 5$.

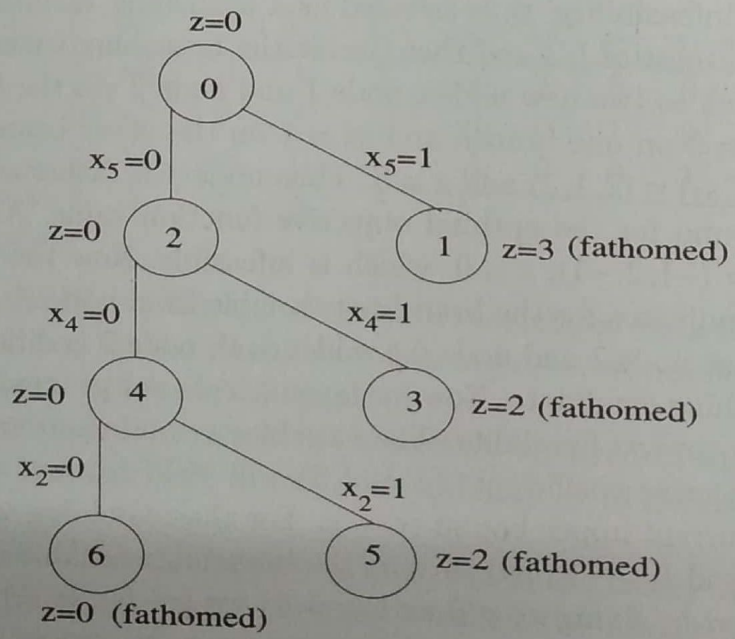


Fig. 6.11.

6.8 Summary and Additional Notes

- Two most basic approaches, namely the cutting plane approach and the branch and bound approach, for solving ILP's are discussed in this chapter.
- R.E.Gomory developed the (fractional) cutting plane method for AILP in 1958 which we present here in section 6.3.
- The cutting plane method for MILP was given by R.E.Gomory in 1960 which we discuss here in section 6.4. The first branch and bound method for solving ILP's was given by A.H.Land, and A. G. Doig in 1960 and the same is presented here in Section 6.5.
- The implicit 0-1 enumerative algorithm discussed here in section 6.7 was given by E. Balas in 1965.
- There are now many other (improved) cutting plane and, branch and bound methods available in the literature. For these developments and other related topics we may refer to certain well known texts on ILP's e.g. Gerfinkel and Nemhauser [65], Hu [78], Salkin [138], Zionts [172] and Taha [155].
- There is all together a different approach for solving ILP's which uses the theory of finite (additive) Abelian groups. The basic work on the group theoretic approach for solving ILP's is due to D.S. Chen who submitted his thesis in the same title in 1970 to the department of Industrial Engineering at SUNY (Buffalo). An elementary exposition of the same is available in Chen and Zionts [36], and Zionts [172].
- Integer programming problems have many applications in capital budgeting, sequencing, and scheduling (e.g. airline crew scheduling). Some famous ILP's are the knapsack problem and the traveling sales person problem. The well known fixed charge problem can be transformed to a ILP. A general separable programming problem can always be approximated by a 0-1 integer linear programming problem. The text by Taha [155] gives details of these applications of ILP's.
- The problems of set covering and matching which occur in the area of the *graph theory* can be modeled as ILP's, e.g. Papadimitriou and Steiglitz [124].

6.9 Exercises

6.1 *Solve the following ILP's graphically*

(1) *Max* $z = x_1 - x_2$

subject to

$$x_1 + 2x_2 \leq 4$$
$$6x_1 + 2x_2 \leq 9$$
$$x_1, x_2 \geq 0$$
x_1 *and* x_2 *integers.*

(2) *Min* $z = 10x_1 + 9x_2$
subject to
$x_1 \leq 8$
$x_2 \leq 10$
$5x_1 + 3x_2 \leq 45$
$x_1, x_2 \geq 0$
x_2 *integer.*

(3) *Max* $z = 4x_1 + 3x_2$
subject to
$3x_1 + 4x_2 \leq 12$
$4x_1 + 2x_2 \leq 9$
$x_1, x_2 \geq 0$
x_1 *and* x_2 *integers.*

(4) *Max* $z = 4x_1 + 3x_2$
subject to
$3x_1 + 4x_2 \leq 12$
$4x_1 + 2x_2 \leq 9$
$x_1, x_2 \geq 0$
x_1 *integer.*

6.2 *Consider the ILP*

Max $z = 15x_1 + 32x_2$
subject to
$7x_1 + 16x_2 \leq 52$
$3x_1 - 2x_2 \leq 9$
$x_1, x_2 \geq 0$
x_1 *and* x_2 *integers.*

1. *Solve the above ILP graphically.*
2. *Show that the optimal solution of the associated LPP (also called the relaxed LPP) is* $(4, 3/2)$.
3. *Does the rounding off of the solution* $(4, 3/2)$ *matches with the optimal solution obtained at (1.) above?*
4. *Solve the given ILP by the Gomory's cutting plane method and also by branch and bound method.*

6.3 *Consider the following AILP*

$$\begin{aligned} &\textit{Max} && z = x_1 + x_2 \\ &\textit{subject to} \\ & && 2x_1 + 5x_2 \le 16 \\ & && 6x_1 + 5x_2 \le 30 \\ & && x_1, x_2 \ge 0 \\ & && x_1 \text{ and } x_2 \text{ integers.} \end{aligned}$$

Let $(LP)_1$ *denote the associated LPP of the given AILP, and the optimal simplex tableau of* $(LP)_1$ *be*

	x_1	x_2	x_3	x_4
$x_2 = 9/5$	0	1	3/10	−1/10
$x_1 = 7/2$	1	0	−1/4	1/4
$Z = 53/10$	0	0	1/20	3/20

1. *Generate the Gomory's cut through the variable* x_2 *and then completely solve the given ILP. Also identify each iteration of the Gomory's cutting plane method graphically.*
2. *Solve the given ILP by branch and bound method.*

6.4 *Solve the following ILP by the Gomory's cutting plane method as well as by the branch and bound method.*

$$\begin{aligned} &\textit{Max} && z = x_1 + x_2 \\ &\textit{subject to} \\ & && 3x_1 + 2x_2 \le 12 \\ & && x_2 \le 2 \\ & && x_1, x_2 \ge 0 \\ & && x_1 \text{ and } x_2 \text{ integers.} \end{aligned}$$

6.5 *Solve the following ILP's by the Gomory's cutting plane method as well as by the branch and bound method.*

$$\begin{aligned} &\textit{Max} && z = x_1 + x_2 \\ &\textit{subject to} \\ & && 2x_1 + 5x_2 \le 16 \\ & && 6x_1 + 5x_2 \le 30 \\ & && x_1, x_2 \ge 0 \\ & && x_1 \text{ integer.} \end{aligned}$$

6.6 *Formulate the following into an equivalent ILP*

$$\begin{aligned} &\textit{Max} && z = 3x_1 + 7x_2 + 5x_3 \\ &\textit{subject to} \\ & && 2x_1 + 5x_2 + 6x_3 \leq 35 \\ & && 5x_1 + 9x_2 \geq 3 \\ & && x_1 = 7 \textit{ or } 13 \textit{ or } 25 \\ & && x_2 \geq 0 \textit{ and integer.} \end{aligned}$$

6.7 *Formulate the mixed integer programming problem equivalent to the following problem*

$$\begin{aligned} &\textit{Max} && z = 3x_1 + 4x_2 \\ &\textit{subject to} \\ & && \textit{either } (x_1 \leq 4 \textit{ and } x_2 \geq 5) \\ & && \textit{or } (x_1 \geq 5 \textit{ and } x_2 \leq 2). \end{aligned}$$

6.8 *Formulate the equivalent mixed integer programming problem for the following problem*

$$\textit{Max} \qquad z = 5x_1 + 6x_2 + 2x_3$$

where (x_1, x_2, x_3) *satisfies at most* 2 *of the following* 4 *constraints*

$$\begin{aligned} x_1 + 2x_2 + x_3 &\leq 5 \\ 3x_1 - 7x_2 + x_3 &\leq 9 \\ 2x_1 + 3x_2 - 7x_3 &\leq 4 \\ -5x_1 + 7x_2 + 8x_3 &\leq 5 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

6.9 *Formulate the equivalent mixed integer programming problem for the following*

$$\begin{aligned} &\textit{Max} && z = 3x_1 + 7x_2 + 5x_3 \\ &\textit{subject to} \\ & && 2x_1 + 5x_2 + 6x_3 \leq (4 \textit{ or } 11 \textit{ or } 17 \textit{ or } 35) \\ & && 5x_1 + 9x_2 \geq 3 \\ & && x_1, x_2 \geq 0 \\ & && x_1 \textit{ and } x_2 \textit{ integers.} \end{aligned}$$

6.10 *Solve the following optimization problem*

$$\begin{aligned} &Max && z = 4x_1 + 3x_2 \\ &subject\ to \\ & && |x_1 + x_2| \geq 2 \\ & && |x_1| \leq 2. \end{aligned}$$

6.11 *Consider the following optimal simplex tableau of the associated LPP for a given AILP (with maximization form and x_3, x_4 as slack variables)*

	x_1	x_2	x_3	x_4
$x_1 = 11/2$	1	0	11/36	−1/36
$x_2 = 9/2$	0	1	−1/12	−1/12
$z = 23/4$	0	0	7/12	1/4

Generate the Gomory's cut constraint through x_1 and hence solve the given AILP. Also show the actual cut constraints graphically.

7
Convex Optimization and Quadratic Programming

7.1 Introduction

A common framework of our studies in earlier chapters has been the presence of *linearity structure* on the given optimization problem, which not only gave beautiful mathematical results but also helped greatly in its algorithmic development. Therefore it may not be wrong to say that the success story of linear programming has its roots in the underlying structure of linearity. However most of the real world applications lead to optimization problems which are inherently nonlinear and therefore are void of linearity structure. Fortunately most often this nonlinearity is of 'parabola' type, leading to the *convexity structure* which can also be exploited to study such nonlinear optimization problems. Our basic aim in this chapter is to understand the *convex optimization problems*, i.e. those optimization problems which have the structure of 'convexity'. These problems are best understood in terms of the convexity/concavity of the objective and constraint functions.

In the later part of this chapter, we also make an algorithmic study of a special type of convex programming problem, namely, the *quadratic programming problem.*

7.2 Convex Functions and their Properties

There is a vast literature on convex sets and convex functions. In the very brief introduction which we wish to present here, we restrict ourselves only to those properties of convex functions which are related to the type of finite dimensional optimization problems introduced in Chapter 1.

Definition 7.2.1. (Convex Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a convex function if for all $x, u \in S$ and for all $0 \leq \lambda \leq 1$, we have*

$$f(\lambda x + (1-\lambda)u) \leq \lambda f(x) + (1-\lambda) f(u). \tag{7.1}$$

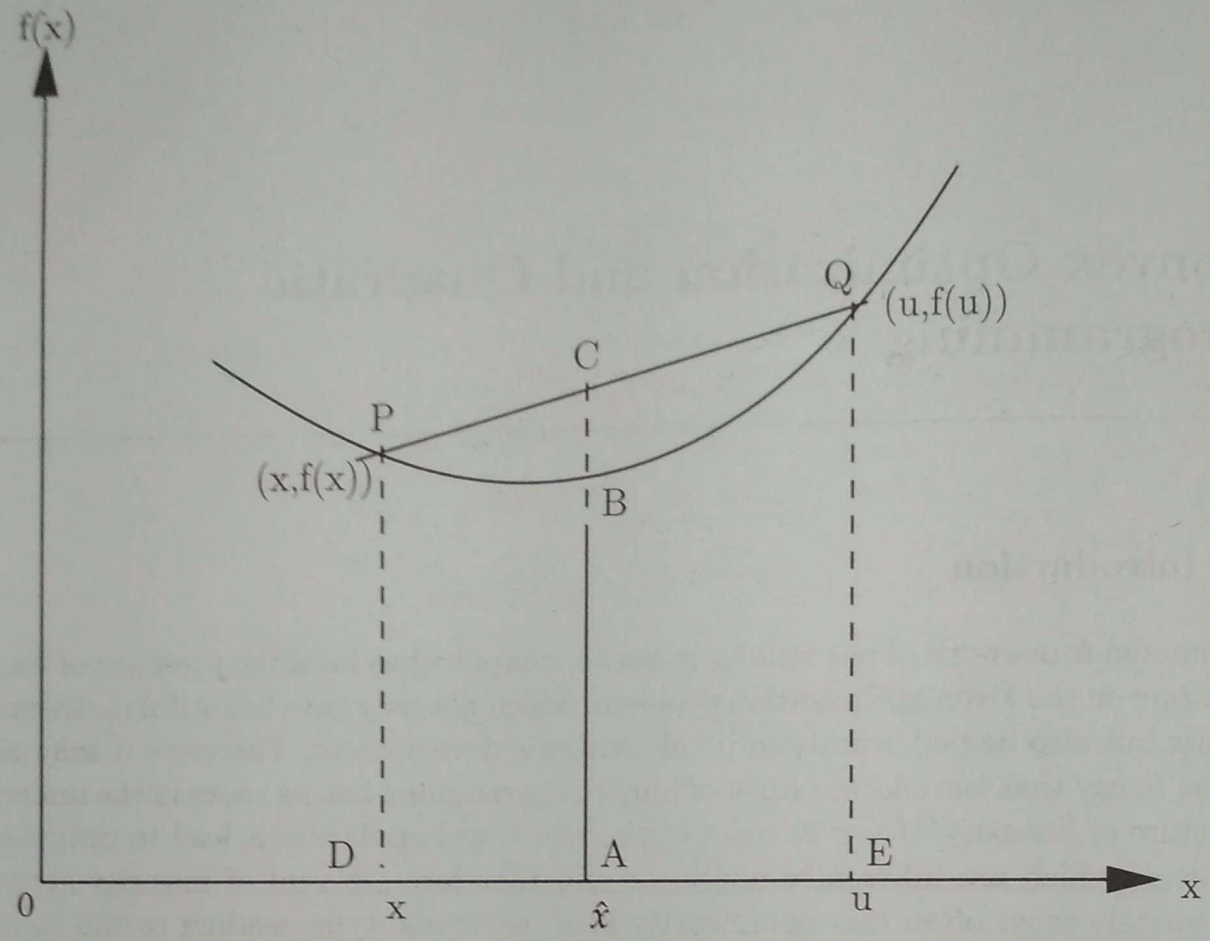


Fig. 7.1.

Let us try to understand the geometrical meaning of the inequality (7.1) by referring to Fig. 7.1.

Here in Fig. 7.1, we have shown the shape of a typical convex function $f : \mathbf{R} \to \mathbf{R}$. Let $\hat{x} = \hat{\lambda}x + (1 - \hat{\lambda})u$ for a particular choice of $\lambda = \hat{\lambda}$, $0 \leq \hat{\lambda} \leq 1$. Then the co-ordinates of the point B are $(\hat{x}, f(\hat{x}))$ and the LHS of inequality (7.1) is the height AB. The RHS of inequality (7.1), being the weighted mean of heights PD and QE with the same weights $\hat{\lambda}$ and $(1 - \hat{\lambda})$, is the height AC. In inequality (7.1), LHS being less than or equal to the RHS, means AB$\leq$ AC. This has to be true for all x, u in the domain and therefore it is possible only when the line segment PQ lies above the graph of the function $y = f(x)$ between P and Q. Therefore a function f is convex if for any two points P and Q on the curve (graph of the function f), the line segment joining P and Q is always on or above the curve between P and Q but never below the curve.

In view of the above discussion it is simple to observe that the following functions are convex functions

(*i*) $f(x) = x^2,\ x \in \mathbf{R}$
(*ii*) $f(x) = |x|,\ x \in \mathbf{R}$
(*iii*) $f(x) = e^x,\ x \in \mathbf{R}$
(*iv*) $f(x) = e^{-x},\ x \in \mathbf{R}$
(*v*) $f(x) = -\sqrt{1-x^2},\ -1 \le x \le 1$.

Definition 7.2.2. (Concave Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a concave function if for all $x,\ u \in S$ and for all $0 \le \lambda \le 1$, we have*

$$f(\lambda x + (1-\lambda)u) \ge \lambda f(x) + (1-\lambda) f(u). \tag{7.2}$$

Again if we take a function $f : \mathbf{R} \to \mathbf{R}$ then the above inequality geometrically means that for all points P and Q on the curve(graph of the function f), the line segment joining P and Q is always on or below the curve between P and Q but never above the curve. The shape of a typical concave function is shown in Fig. 7.2.

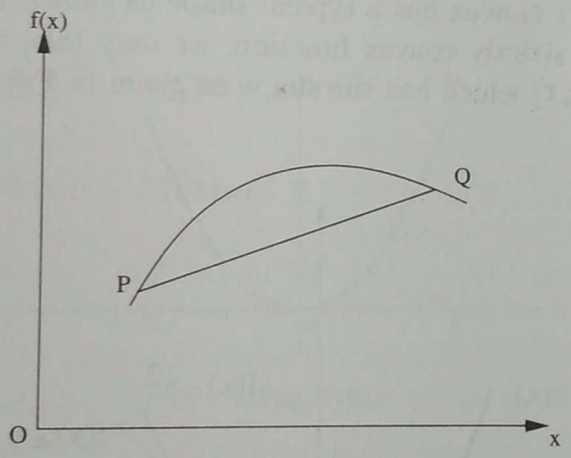


Fig. 7.2.

From the above discussion, it is obvious that f is a concave function if and only if $-f$ is a convex function.

Therefore we can check that the following are concave functions

(*i*) $f(x) = \ln x,\quad x > 0$
(*ii*) $f(x) = +\sqrt{1-x^2},\quad -1 \le x \le 1$
(*iii*) $f(x) = -|x|,\quad x \in \mathbf{R}$.

If the inequality (7.1) is strict then the function f is called a *strictly convex function*. Thus every strictly convex function is convex but the converse is not true. Similarly if the inequality (7.2) is strict then the function f is called a *strictly concave function*.

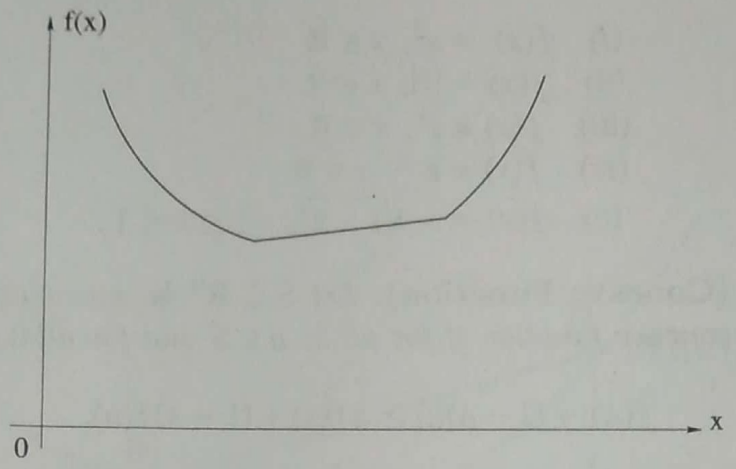


Fig. 7.3.

The examples of convex functions (respectively concave functions) given above are really strictly convex functions (respectively strictly concave functions). A convex function which is not strictly convex has a typical shape as shown in Fig 7.3

As an example of a strictly convex function, we may take the function $\phi : \mathbf{R} \to \mathbf{R}$ given by $\phi(x) = Max(x^2, x)$ which has the shape as given in Fig. 7.4.

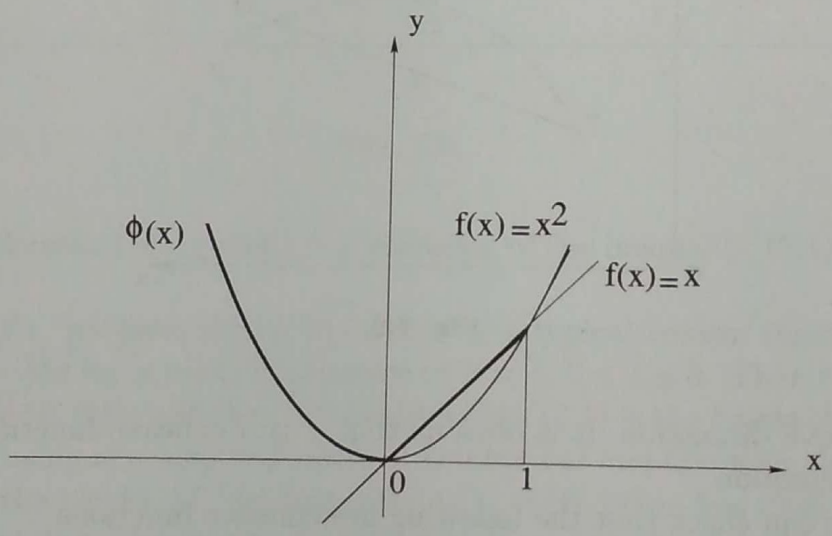


Fig. 7.4.

From the above examples, we make note of the following

1. If a function is both convex and concave, then it has to be a linear function.
2. A function may be neither convex nor concave, e.g. $f(x) = \sin x$, $\frac{-\pi}{2} \leq x \leq \frac{\pi}{2}$ or $f(x) = x^3$, $x \in \mathbf{R}$.
3. The domain of a convex function has to be a convex set. This is because in the L.H.S. of (7.1) we have to evaluate the function at $\hat{x} = \lambda x + (1 - \lambda)u$ for all $x, u \in S$

and for all $0 \leq \lambda \leq 1$, so $\hat{x}$ has to be in the domain S. Obviously the domain of a concave function has also to be convex set.

4. A convex/concave function need not be differentiable; e.g. $f(x) = |x|$, $x \in \mathbf{R}$ is convex but is not differentiable at $x = 0$.
5. A convex function need not even be continuous, e.g.

$$f(x) = \begin{cases} x^2, & -1 \leq x < 1 \\ 2, & x = 1 \end{cases}$$

is not continuous at $x = 2$ as can be seen from Fig 7.5. However, we can show that a convex function is always continuous in the interior of its domain. Thus if f is convex over a convex set $S \subseteq \mathbf{R}^n$, then the points of discontinuity (if any) could only be on the boundary of S.

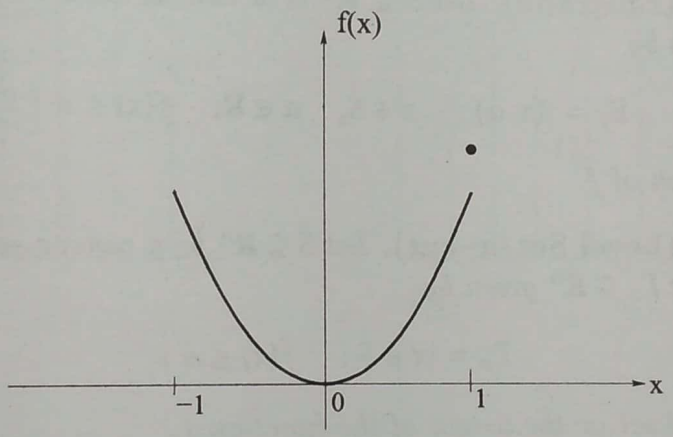


Fig. 7.5.

6. Let f and g be two convex functions defined over a convex set $S \subseteq \mathbf{R}^n$ then (a) $f+g$, (b) αf $(\alpha \geq 0)$, and (c) $h(x) = \underset{x \in S}{Max}\,(f(x), g(x))$ are also convex functions. Thus $-x + 2x^2 + |x|$ is a convex function for $x \in \mathbf{R}$. Also $h(x) = Max(x, x^2)$, $x \in \mathbf{R}$ is a convex function (see Fig. 7.4).
7. Results similar to (6) above obviously hold for the concave functions as well. Thus if f and g are concave functions defined over a convex set $S \subseteq \mathbf{R}^n$ then so are the functions $f + g$, αf, $\alpha \geq 0$ and $h(x) = \underset{x \in S}{Min}(f(x), g(x))$. As an example, let $f(x) = +\sqrt{1 - x^2}$, $0 \leq x \leq 1$ and $g(x) = +\sqrt{x}$, $0 \leq x \leq 1$. Then the min function h is a concave function (see Fig. 7.6). Also the function $-x - 2x^2 + \ln x$, $(x > 0)$ is a concave function.

In the study of convex/concave functions, certain sets play an important role. These are the *epigraph*, the *α−cut* or the *level set* and the *hypograph*.

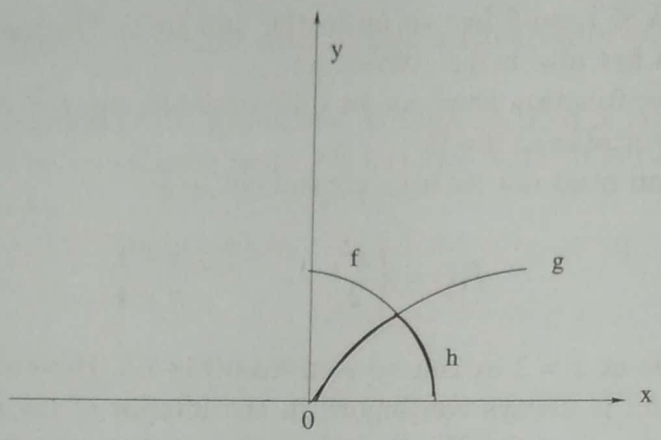


Fig. 7.6.

Definition 7.2.3. (Epigraph). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then the set $E_f \subseteq \mathbf{R}^{n+1}$ given by*

$$E_f = \{(x, \alpha) : \quad x \in S, \quad \alpha \in \mathbf{R}, \quad f(x) \leq \alpha \}$$

is called the epigraph of f.

Definition 7.2.4. (Level Set/α−cut). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Let $\alpha \in \mathbf{R}$. Then the set $\Gamma_\alpha \subseteq \mathbf{R}^n$ given by*

$$\Gamma_\alpha = \{x \in S : \quad f(x) \leq \alpha \}$$

is called the α−level set or the α−cut of the function f.

Definition 7.2.5. (Hypograph). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then the set $G_f \subseteq \mathbf{R}^{n+1}$ given by*

$$G_f = \{(x, \alpha) : \quad x \in S, \quad \alpha \in \mathbf{R}, \quad f(x) \geq \alpha \}$$

is called the hypograph of f.

A visualization of these sets for $S \subseteq \mathbf{R}$ is given in Fig. 7.7. We now have the following results.

Theorem 7.2.1 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is a convex function on S if and only if its epigraph E_f is a convex set.*

Proof. (i) (Necessity) Let f be convex on S and (x, α) and $(u, \beta) \in E_f$. Then by the convexity of f on S, we have that for $0 \leq \lambda \leq 1$,

$$\begin{aligned} f(\lambda x + (1-\lambda)u) &\leq \lambda f(x) + (1-\lambda) f(u) \\ &\leq \lambda\alpha + (1-\lambda)\beta \,. \end{aligned}$$

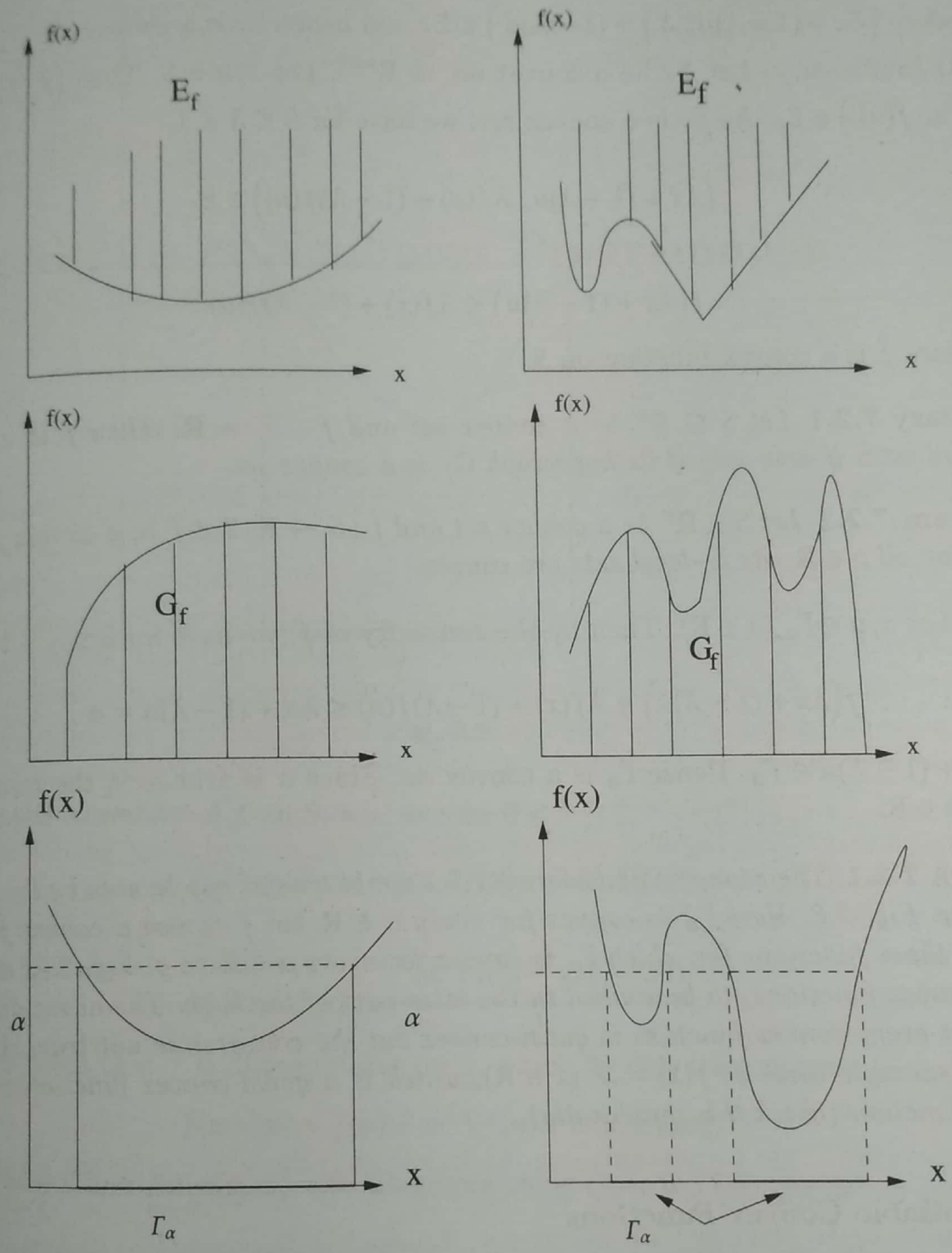


Fig. 7.7.

Therefore $\big(\lambda x + (1-\lambda)u,\ \lambda\alpha + (1-\lambda)\beta\ \big) \in E_f$ and hence E_f is a convex set.

(ii) (Sufficiency) Let E_f be a convex set in $\mathbf{R}^{n+1}$. Let $x, u \in S$. Then $\big(x,\ f(x)\ \big) \in E_f$ and $\big(u,\ f(u)\ \big) \in E_f$. As E_f is a convex set, we have for $0 \le \lambda \le 1$,

$$\big(\lambda x + (1-\lambda)u,\ \lambda f(x) + (1-\lambda)f(u)\big) \in E_f$$

i.e.

$$f\big(\lambda x + (1-\lambda)u\big) \le \lambda f(x) + (1-\lambda)f(u).$$

Therefore f is a convex function on S. □

Corollary 7.2.1 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is a concave function on S if and only if its hypograph G_f is a convex set.*

Theorem 7.2.2 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Let f be a convex function. Then for all $\alpha \in \mathbf{R}$, its α−level sets are convex.*

Proof. Let $x, u \in \Gamma_\alpha$ $(\alpha \in \mathbf{R})$. Then by the convexity of f, we have for $0 \le \lambda \le 1$

$$f\big(\lambda x + (1-\lambda)u\big) \le \lambda f(x) + (1-\lambda)f(u) \le \lambda\alpha + (1-\lambda)\alpha = \alpha\ ,$$

i.e. $\lambda x + (1-\lambda)u \in \Gamma_\alpha$. Hence Γ_α is a convex set. Since α is arbitrary, the result holds for all $\alpha \in \mathbf{R}$. □

Remark 7.2.1 *The converse of Theorem 7.2.2 is not true as can be seen by the example shown in Fig. 7.8. Here Γ_α is convex for every $\alpha \in \mathbf{R}$ but f is not a convex function. In fact, those functions for which Γ_α is convex for every $\alpha \in \mathbf{R}$ (e.g. Fig. 7.8) are called quasi-convex functions, to be studied in the later part of the book. The above discussion tells that every convex function is quasi-convex but the converse is not true. A specific counter example could be $f(x) = x^3$ $(x \in \mathbf{R})$, which is a quasi-convex function but not a convex function (check this graphically).*

Differentiable Convex Functions

We now present some of the properties of differentiable and twice differentiable convex functions.

Theorem 7.2.3 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be differentiable. Let f be a convex function on S. Then for all $x, u \in S$, we have*

$$f(x) - f(u) \ge (x-u)^T \nabla f(u). \tag{7.3}$$

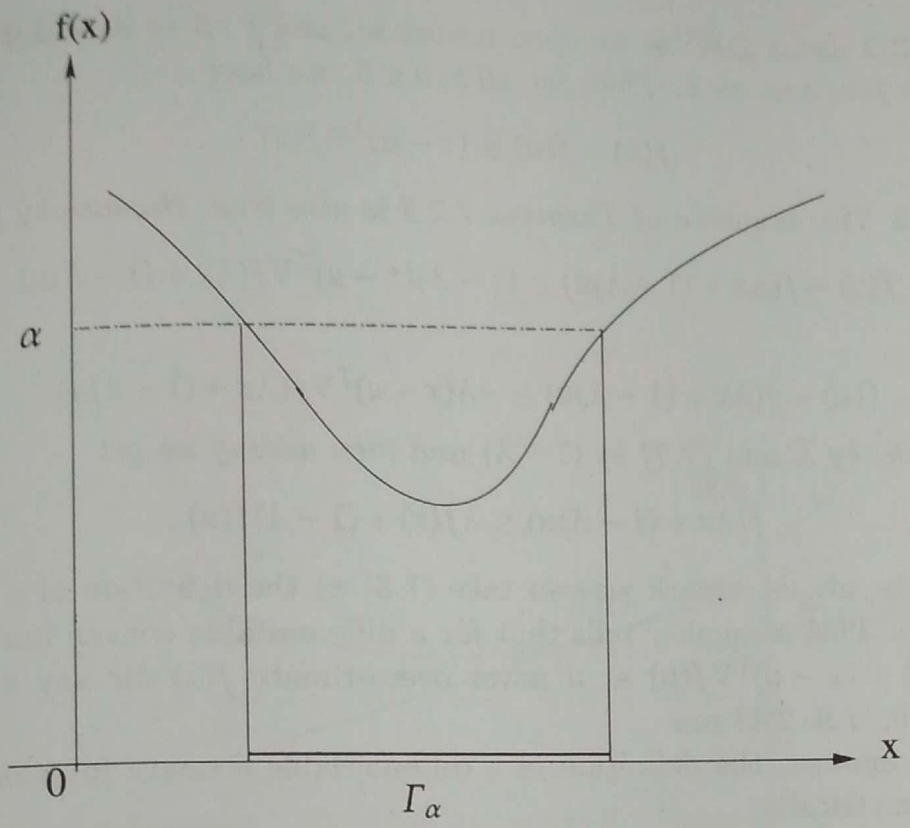


Fig. 7.8.

Proof. By the convexity of f on S, we have for $0 \le \lambda \le 1$,

$$f\big(\lambda x + (1-\lambda)u\big) \le \lambda f(x) + (1-\lambda) f(u)\,,$$

i.e.

$$f(x) - f(u) \ge \frac{f\left(u + \lambda(x-u)\right) - f(u)}{\lambda}. \tag{7.4}$$

But we are given that f is differentiable on S, which by definition means

$$f(u+w) = f(u) + w^T \nabla f(u) + \alpha(u,w)\|w\|, \tag{7.5}$$

where $u + w \in S$ and $\lim_{w \to 0} \alpha(u,w) = 0$. Therefore using (7.5) in (7.4), we get

$$f(x) - f(u) \ge \frac{f(u) + \lambda(x-u)^T \nabla f(u) + \alpha(u, \lambda(x-u))\lambda\|x-u\| - f(u)}{\lambda},$$

i.e.

$$f(x) - f(u) \ge (x-u)^T \nabla f(u) + \alpha(u, \lambda(x-u))\|x-u\|.$$

Now on taking the limit as $\lambda \to 0$, we get $\lim_{\lambda \to 0} \alpha(u, \lambda(x-u)) = 0$, and hence

$$f(x) - f(u) \ge (x-u)^T \nabla f(u)$$

as required. □

Corollary 7.2.2 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be differentiable. Let f be a concave function on S. Then for all $x, u \in S$, we have*

$$f(x) - f(u) \leq (x-u)^T \nabla f(u).$$

Remark 7.2.2 *The converse of Theorem 7.2.3 is also true. Because by (7.3) we have*

$$f(x) - f(\lambda x + (1-\lambda)u) \geq (1-\lambda)(x-u)^T \nabla f(\lambda x + (1-\lambda)u) \tag{7.6}$$

and

$$f(u) - f(\lambda x + (1-\lambda)u) \geq -\lambda(x-u)^T \nabla f(\lambda x + (1-\lambda)u)\,. \tag{7.7}$$

Multiplying (7.6) by λ and (7.7) by $(1-\lambda)$ and then adding we get

$$f(\lambda x + (1-\lambda)u) \leq \lambda f(x) + (1-\lambda) f(u)\,.$$

In view of the above remark we can take (7.3) as the definition of a differentiable convex function. This inequality tells that for a differentiable convex function, the linearlization $f(u) + (x-u)^T \nabla f(u)$ at u never overestimate $f(x)$ for any $x \in S$. This is illustrated in Fig. 7.9. 2:43 pm

In a similar manner, the definition of a differentiable concave function can also be interpreted geometrically.

Theorem 7.2.4 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be differentiable. Then f is a convex function on S if and only if for all $x, u \in S$, we have*

$$(x-u)^T [\nabla f(x) - \nabla f(u)] \geq 0. \tag{7.8}$$

Proof. (i) (Necessity). For $x, u \in S$, we have from Theorem 7.2.3

$$f(x) - f(u) - (x-u)^T \nabla f(u) \geq 0,$$

and

$$f(u) - f(x) - (u-x)^T \nabla f(x) \geq 0.$$

Adding these two inequalities, we get

$$(x-u)^T [\nabla f(x) - \nabla f(u)] \geq 0.$$

(ii) (Sufficiency.) Let $x, u \in S$ and $0 \leq \lambda \leq 1$. Then by the mean value theorem, we have

$$f(x) - f(u) = (x-u)^T \nabla f(u + \overline{\lambda}(x-u)), \quad \text{for some } 0 < \overline{\lambda} < 1\,. \tag{7.9}$$

Also by assumption

$$\overline{\lambda}(x-u)^T [\nabla f\big(u + \overline{\lambda}(x-u)\big) - \nabla f(u)] \geq 0\,.$$

But then (7.9) gives

$$f(x) - f(u) \geq (x-u)^T \nabla f(u)\,,$$

as required.

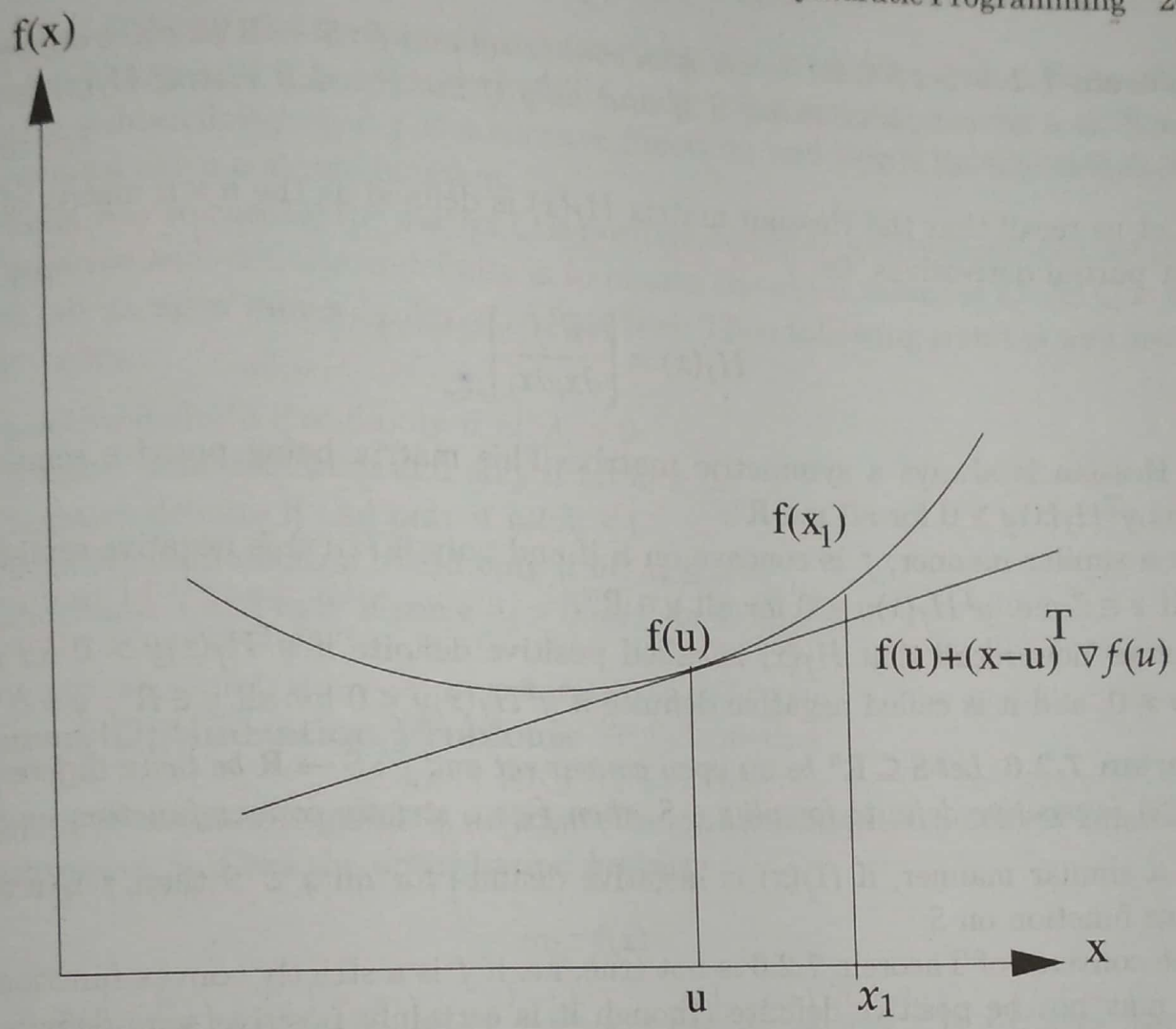


Fig. 7.9.

Remark 7.2.3 *For a concave differentiable function, the inequality (7.8) becomes*

$$(x-u)^T[\nabla f(x)-\nabla f(u)] \leq 0.$$

Remark 7.2.4 *If $f : \mathbf{R} \to \mathbf{R}$, then inequality (7.8) means*

$$(x-u)\big(f'(x)-f'(u)\big) \geq 0,$$

i.e. for $x \geq u$, $f'(x) \geq f'(u)$. Thus f' is an increasing function which is the well known definition of convexity for a real valued function of real variable. So the inequality (7.8) can be seen as an extension of this basic result to $\mathbf{R}^n$.

Though we shall not prove here, results similar to Theorems 7.2.3 and 7.2.4 can also be stated for strictly convex and strictly concave functions with the obvious modification that all inequalities are strict.

We next assume that f is twice differentiable on S. Then we state the following theorem which we do not prove here.

Theorem 7.2.5 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be twice differentiable. Then f is a convex function on S if and only if the Hessian matrix $H_f(x)$ is positive semi-definite for all $x \in S$.*

Let us recall that the Hessian matrix $H_f(x)$ is defined as the $n \times n$ matrix of second order partial derivatives, i.e.

$$H_f(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n\times n}.$$

The Hessian is always a symmetric matrix. This matrix being positive semi-definite means $y^T H_f(x) y \geq 0$ for all $y \in \mathbf{R}^n$.

In a similar manner, f is concave on S if and only if $H_f(x)$ is negative semi-definite for all $x \in S$, i.e. $y^T H_f(x) y \leq 0$ for all $y \in \mathbf{R}^n$.

Let us now recall that $H_f(x)$ is called positive definite if $y^T H_f(x) y > 0$ for all $y \in \mathbf{R}^n$, $y \neq 0$, and it is called negative definite if $y^T H_f(x) y < 0$ for all $y \in \mathbf{R}^n$, $y \neq 0$.

Theorem 7.2.6 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be twice differentiable. If $H_f(x)$ is positive definite for all $x \in S$, then f is a strictly convex function on S.*

In a similar manner, if $H_f(x)$ is negative definite for all $x \in S$ then f is a strictly concave function on S.

The converse of Theorem 7.2.6 is not true, i.e. if f is a strictly convex function then, $H_f(x)$ may not be positive definite (though it is certainly positive semi-definite), e.g. $f(x) = x^4$, $x \in \mathbf{R}$ is a strictly convex function on $\mathbf{R}$ but $H_f(x) = 12x^2$ is not positive definite for $x = 0$.

Example 7.2.1 *Examine the convexity/strict-convexity of the functions (i) $f(x_1, x_2) = 2x_1^2 + x_2^2 + 4x_1x_2$ and (ii) $4x_1^2 + x_2^2 + 4x_1x_2$.*

Solution. (i) $f(x_1, x_2) = 2x_1^2 + x_2^2 + 4x_1x_2$ gives the Hessian matrix as

$$\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

which is positive definite. Hence f is a strictly convex function.

(ii) $f(x_1, x_2) = 4x_1^2 + x_2^2 + 4x_1x_2$ gives the Hessian matrix as

$$\begin{bmatrix} 8 & 4 \\ 4 & 2 \end{bmatrix}$$

which is positive semi-definite. Hence f is a convex function.

In general if $f : \mathbf{R}^n \to \mathbf{R}$ is a quadratic form, i.e. $f(x) = x^T Q x$, where Q is real symmetric. Then we can check that $\nabla f(x) = 2Qx$ and $H_f(x) = 2Q$. Therefore the nature of the given quadratic form $x^T Q x$ will depend upon the nature of the matrix Q. If Q is

positive definite, then f is a strictly convex function; if it is positive semi-definite, then f is a convex function; if it is negative definite then f is a strictly concave function; if it is negative semi-definite, then f is a concave function; and if it is indefinite then f is neither a convex nor a concave function.

The easiest way to check if the matrix Q is positive definite/positive semi-definite/negative definite/negative semi-definite/indefinite is to obtain the eigen values of Q. As Q is real symmetric, all its eigen values $\lambda_1, \lambda_2, \ldots, \lambda_n$ are real. Then following result is well known in matrix theory

1. Q is positive definite if and only if *all* $\lambda_i > 0$.
2. Q is positive semi definite if and only if *all* $\lambda_i \geq 0$.
3. Q is negative definite if and only if *all* $\lambda_i < 0$.
4. Q is negative semi-definite if and only if *all* $\lambda_i \leq 0$.
5. Q is indefinite if and only if some $\lambda_i > 0$ and some $\lambda_i < 0$.

7.3 Convex Optimization Problems

By a *convex optimization problem* we mean the minimization of a convex function f over a convex set S. Thus the optimization problem

$$\min_{x \in S} f(x) \tag{7.10}$$

is a convex optimization problem if $S \subseteq \mathbf{R}^n$ is a convex set and f is a convex function on S.

If the problem is given in the maximization form i.e. $\max f(x)$, subject to $x \in S$ then the problem will also be called as a convex optimization problem if $S \subseteq \mathbf{R}^n$ is a convex set and f is a concave function on S.

We now present certain basic results with regard to the convex optimization problem (7.10), the analogous results for the maximization case hold with obvious modification of changing the convexity of f to the concavity of f.

Theorem 7.3.1 *Let $\overline{x}$ be a local min point of the convex optimization problem (7.10). Then $\overline{x}$ is also its global min point.*

Proof. We are given that $\overline{x}$ is a local min point of (7.10). This, by definition, implies that there exists $\delta > 0$, such that $f(\overline{x}) \leq f(u)$ for all $u \in N_\delta(\overline{x}) \cap S$, where $N_\delta(\overline{x})$ is a ball of radius δ centered at $\overline{x}$.

Let x be an arbitrary point outside the set $N_\delta(\overline{x}) \cap S$. The theorem will then be proved if we show that $f(\overline{x}) \leq f(x)$. Now, there certainly exists $\overline{\lambda}$, $0 < \overline{\lambda} < 1$ such that $x^* = \overline{\lambda}x + (1 - \overline{\lambda})\overline{x}$ and $x^* \in N_\delta(\overline{x}) \cap S$. The Fig. (7.10) makes this assertion quite clear.

As $x^* \in N_\delta(\overline{x}) \cap S$, and $\overline{x}$ is a local min point of (7.10) we have $f(\overline{x}) \leq f(x^*)$. This gives

$$f(\overline{x}) \leq f(\overline{\lambda}x + (1 - \overline{\lambda})\overline{x}\,) .$$

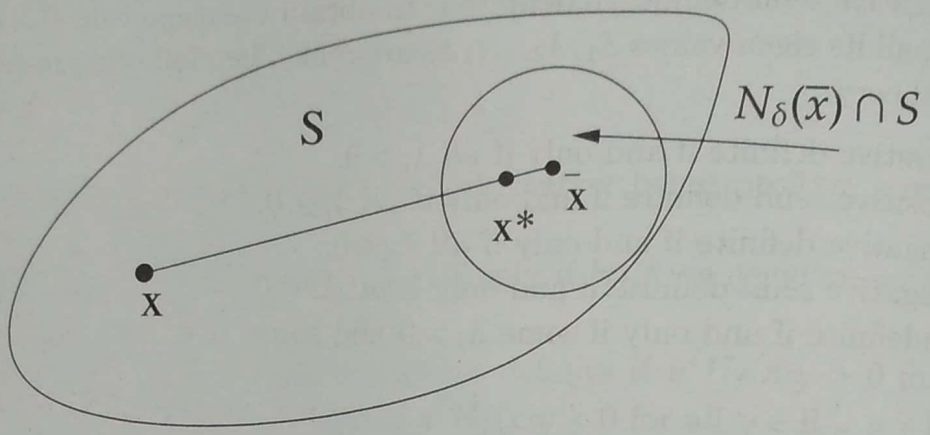


Fig. 7.10.

But f is convex on S, and hence the above gives

$$f(\bar{x}) \leq \bar{\lambda} f(x) + (1 - \bar{\lambda}) f(\bar{x}),$$

i.e.

$$\bar{\lambda} f(\bar{x}) \leq \bar{\lambda} f(x),$$

i.e.

$$f(\bar{x}) \leq f(x) \text{ as } 0 < \bar{\lambda} < 1;$$

which implies that $\bar{x}$ is a global min point. □

Theorem 7.3.2 *The set of all optimal solutions of the convex optimization problem (7.10) is a convex set.*

Proof. Let the set of all optimal solutions of problem (7.10) be denoted by V, i.e.

$$V = \{\bar{x} : \bar{x} \in S, \ f(\bar{x}) \leq f(x) \text{ for all } x \in S\}.$$

Let $u, w \in V$. Then $\hat{x} = \lambda u + (1 - \lambda) w$, $0 \leq \lambda \leq 1$ certainly belong to S because S is a convex set. Further by the convexity of f

$$f(\hat{x}) = f(\lambda u + (1 - \lambda) w) \leq \lambda f(u) + (1 - \lambda) f(w). \tag{7.11}$$

But $f(u) \leq f(x)$ and $f(w) \leq f(x)$ for all $x \in S$, as $u, w \in V$. Hence by (7.11),

$$f(\hat{x}) \leq \lambda f(x) + (1 - \lambda) f(x),$$

i.e.

$$f(\hat{x}) \leq f(x) \quad \text{for all } x \in S.$$

Thus $\hat{x} \in S$ is also optimal for (7.10), i.e. $\hat{x} \in V$. Therefore V is a convex set. □

Theorem 7.3.3 *Let a nonconstant convex function f be maximized over a convex set S. Then no interior point of S can be a maximizing point; i.e. if a maximizing point exists, then it must be a boundary point of S.*

Proof. If there is no maximizing point of f over S, then the assertion is vacuously true. Let us therefore assume that f has a maximizing point x^* over S. As f is nonconstant and x^* is a maximizing point, there certainly exists a point $x \in S$ such that $f(x^*) > f(x)$.

Let z be an arbitrary point in the interior of S. The theorem will be proved if we can show that z can never be a maximizing point of f over S. By the definition of interior point, there exists $\delta > 0$ such that $N_\delta(z) \subseteq S$. Then as the below given figure illustrates, there is a point $y \in S$ and $\overline{\lambda}$, $0 < \overline{\lambda} \leq 1$ such that $z = \overline{\lambda}x + (1 - \overline{\lambda})y$.

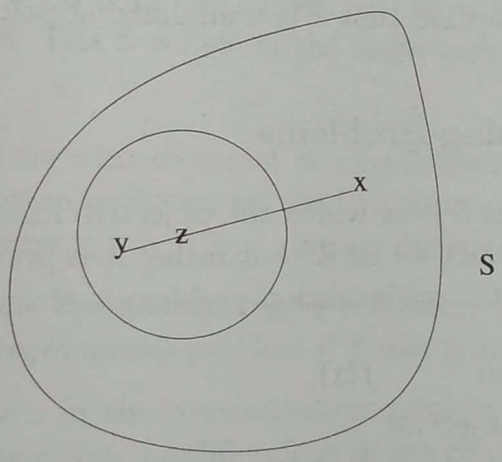


Fig. 7.11.

Now using the convexity of f over S, we get

$$\begin{aligned} f(z) &= f(\overline{\lambda}x + (1 - \overline{\lambda})y) \\ &\leq \overline{\lambda} f(x) + (1 - \overline{\lambda}) f(y) \\ &< \overline{\lambda} f(x^*) + (1 - \overline{\lambda}) f(x^*), \end{aligned} \tag{7.12}$$

because for the choice of x, $f(x) < f(x^*)$ and $f(y) \leq f(x^*)$, x^* being a maximizing point. But then (7.12) implies $f(z) < f(x^*)$, and hence z cannot be a maximizing point of f over S. □

Remark 7.3.1 *In Theorem 7.3.1, if S is a polytope then we can additionally say that at least one corner point is a maximizing point. Thus if we are maximizing (respectively*

minimizing) a convex function (respectively concave function) over a polytope S, then at least one corner point of the polytope is optimal.

Theorem 7.3.4 *Let $S \subseteq \mathbf{R}^n$ be convex and $f : S \to \mathbf{R}$ be a strictly convex function. Then there is a unique minimizing point of f over S.*

Proof. Let $\bar{x}$ and x^* $(\bar{x} \neq x^*)$ be two minimizing points of f over S, i.e. $\bar{x} \in S$, $x^* \in S$, $f(\bar{x}) \geq f(x)$ for all $x \in S$, $f(x^*) \geq f(x)$ for all $x \in S$, and $f(\bar{x}) = f(x^*)$.

Let $\hat{x} = \lambda\bar{x} + (1-\lambda)x^*$ for $0 < \lambda < 1$. Then by the strict convexity of f we have

$$f(\hat{x}) = f(\lambda\bar{x} + (1-\lambda)x^*) < \lambda f(\bar{x}) + (1-\lambda)f(x^*),$$

i.e.

$$\lambda f(\hat{x}) < \lambda f(\bar{x}),$$

i.e.

$$f(\hat{x}) < f(\bar{x}),$$

which contradicts the assumption that $\bar{x}$ is a minimizing point. □

7.4 Convex Programming Problems

In many applications the set S over which the objective function f is to be minimized or maximized is not an abstract set in $\mathbf{R}^n$ but rather it is prescribed by a finite number of constraint functions; i.e. the optimization problem is of the form

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \leq 0 \quad (i = 1, 2, \ldots, m). \end{aligned} \tag{7.13}$$

These problems are called *mathematical programming problems* and, as discussed in Chapter 1, are classified into two broad classes, namely *linear programming problems* and *nonlinear programming problems.*

While discussing the mathematics of the simplex method in Chapter 3, we noted that we could develop simplex method only because of the linearity structure of LPP's. This linearity structure gave three very important properties for LPP's, namely (P1) the feasible region is a convex set, (P2) if the given LPP has an optimal solution then at least one corner point is optimal, and (P3) every local optimal point is also a global optimal point. There, through certain examples, we also observed that, in general, there is no guarantee that for nonlinear programming problems any of these properties hold and therefore it may not be possible to develop algorithms which could solve every type of nonlinear programming problems.

A class of nonlinear programming problems for which some of the above properties of LPP's continue holding is the class of *convex programming problems*. Basically a convex programming problem is a problem of form (7.13) where we prescribe appropriate convexity/concavity requirements on f and g_i $(i = 1,2,\ldots,m)$ so that we end up with minimizing a convex function over a convex set, i.e. it becomes a convex optimization problem. Let S denote the feasible region of (7.13), i.e.

$$S = \{\, x \in \mathbf{R}^n : \; g_i(x) \leq 0 \,,\; (i = 1,2,\ldots,m) \,\}.$$

Theorem 7.4.1 *Let for each $i = 1,2,\ldots,m$, g_i be a convex function. Then S is a convex set.*

Proof. Let S_i be the set of points $x \in \mathbf{R}^n$ for which the ith constraint holds, i.e.

$$S_i = \{\, x \in \mathbf{R}^n : \; g_i(x) \leq 0 \,\}.$$

Now S_i can also be identified as a $\alpha - cut$ or α–level set of the function g_i for $\alpha = 0$. As g_i is a convex function, all its α–cuts are convex set; so in particular 0-cut is a convex set. Thus S_i is a convex set. But $S = \cap_i S_i$ is the finite intersection of convex sets and hence it is a convex set. □

In view of the above, if we wish to recast the mathematical programming problem (7.13) as a convex optimization problem, we need to ensure that the objective function f and the constraint functions g_i $(i = 1,2,\ldots,m)$ are convex functions.

Definition 7.4.1 (Convex Programming Problem). *The optimization problem (7.13) is called a convex programming problem if f and g_i $(i = 1,2,\ldots,m)$ are convex.*

If problem (7.13) is given in the maximization form or the constraints are in '$\geq$' form then we can make appropriate modifications in the convexity requirements so that it becomes a convex programming problem. Let us remember that the bottom line is that we should minimize a convex function over a convex set. The following table is self explanatory.

Min $f(x)$ subject to $g_i(x) \leq 0 \quad (i = 1,2,\ldots,m).$ (f, g_i are convex)	Max $f(x)$ subject to $g_i(x) \leq 0 \quad (i = 1,2,\ldots,m).$ (f is concave , g_i are convex)
Min $f(x)$ subject to $g_i(x) \geq 0 \quad (i = 1,2,\ldots,m).$ (f is convex , g_i are concave)	Max $f(x)$ subject to $g_i(x) \geq 0 \quad (i = 1,2,\ldots,m).$ (f, g_i are concave)

Example 7.4.1 *Check if the following problem*

$$\begin{aligned} &Max \quad 4x_1 + 3x_2 \\ &subject\ to \\ &\quad x_1 + x_2 \leq 4 \\ &\quad x_1 x_2 \leq 1 \\ &\quad x_1,\ x_2 \geq 0 \end{aligned}$$

is a convex programming problem.

Solution Here $f(x_1, x_2) = 4x_1 + 3x_2$, $g_1(x_1, x_2) = (x_1 + x_2 - 4)$ and $g_2(x_1, x_2) = x_1 x_2 - 1$, $g_3(x_1, x_2) = -x_1$, *and* $g_4(x_1, x_2) = -x_2$. Since f is a linear function which is being maximized it can certainly be taken as a concave function. Also the functions g_1, g_3 and g_4 are convex as they are linear functions of (x_1, x_2). So we have to check the convexity of the function g_2 only. For this we compute its Hessian and get

$$H_{g_2}(x_1, x_2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which is not positive semi-definite as its eigen values are $+1$ and -1. Thus g_2 is not a convex function and so the given problem is not a convex programming problem. We are checking about the convexity of g_1, g_2 g_3, and g_4 because we have taken the constraints in the '$\leq$' form for defining g_1, g_2 g_3, and g_4 here.

In view of the above, there is no guarantee that the feasible region S is a convex set. In fact it is really not a convex set as can be verified by plotting the given constraints as shown in Fig. 7.12

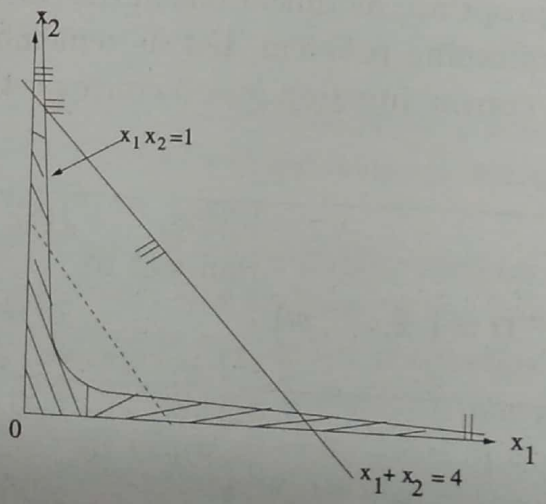


Fig. 7.12.

Example 7.4.2 *Check if the following problem is a convex programming problem.*

$$\begin{aligned} &Max && x_2 \\ &subject\ to \\ &&& x_1^2 + x_2^2 \leq 1 \\ &&& x_1^2 \geq x_2. \end{aligned}$$

Solution. It is a maximization problem and the objective function is linear so certainly concave. The constraint functions are $g_1(x_1, x_2) = x_1^2 + x_2^2 - 1$ and $g_2(x_1, x_2) = -x_1^2 + x_2$ with

$$H_{g_1}(x_1, x_2) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad H_{g_2}(x_1, x_2) = \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Here H_{g_1} is positive definite but H_{g_2} is negative semi-definite; and hence g_2 is not a convex function. Therefore the given problem is not a convex programming problem. In this case again we get a non-convex feasible region S as illustrated in Fig 7.13

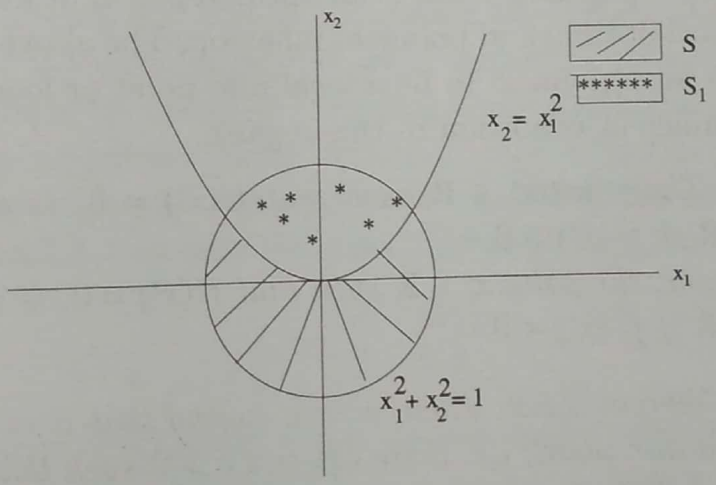


Fig. 7.13.

In the above problem, if we change the second constraint to $x_1^2 \leq x_2$, then $\hat{g}_2(x_1, x_2) = x_1^2 - x_2$ comes out to be a convex function. This makes this new problem a convex programming problem whose feasible region S_1 is a convex set.

Remark 7.4.1 *If the feasible region S given by the constraints $g_i(x) \leq 0$ $(i = 1, 2, \ldots, m)$ is not a convex set, then some g_i must certainly be a non-convex function. However, if the feasible region given by the above constraints is a convex set, it does not necessarily mean that every g_i is a convex function; in fact some g_i can very well be a quasi-convex function because the convexity of $\alpha-$cuts is a characterizing property of a quasi-convex function.*

7.5 Optimality Conditions: Motivations from Elementary Calculus

In this section we shall try to write optimality conditions for the nonlinear programming problem (7.13) taking clue from what we have studied earlier in the calculus course for minimizing/maximizing functions of one and two variables. As such, we do not plan to give any mathematical proof here because optimality conditions for NLP's are studied in detail later in Chapter 9.

Let us start with the unconstrained minimization/maximization of functions of one variable, i.e. the problem of the form 'min $f(x)$ over $x \in \mathbf{R}$' or 'max $f(x)$ over $x \in \mathbf{R}$'. We assume that f is twice continuously differentiable over $\mathbf{R}$. The following are the standard results in calculus.

Theorem 7.5.1 *If $x^* \in \mathbf{R}$ is a local min or local max point of f over $\mathbf{R}$ then $f'(x^*) = 0$.*

Geometrically, the above theorem tells that at an optimal point (which may be local or global) the tangent line is parallel to the $x-$axis. If we agree to call those points x for which $f'(x) = 0$, as the *critical points or the stationary points* of the function f, then this implies that every optimal point of f over $\mathbf{R}$ is a stationary point.

Obviously not every stationary point is an optimal point. Stationary points of f over $\mathbf{R}$ include points of local optima and points of inflexion. The above theorem is essentially a necessary condition for a point x^* to be a local min point or local max point of f over $\mathbf{R}$. We now state a sufficient condition in this regard.

Theorem 7.5.2 *(i) The point $x^* \in \mathbf{R}$ such that $f'(x^*) = 0$, is an unconstrained local min point of f over $\mathbf{R}$ if $f''(x^*) > 0$.*
(ii) In a similar manner, the point $x^* \in \mathbf{R}$ such that $f'(x^*) = 0$, is an unconstrained local max point of f over $\mathbf{R}$ if $f''(x^*) < 0$.

Remark 7.5.1 *In Theorem 7.5.2, when we are saying that it is a local min point, we really mean strict local min point; i.e. there exists a $\delta > 0$ such that $f(x^*) < f(x)$ for all x satisfying $|x - x^*| < \delta$, $x \neq x^*$. Similarly the point x^* satisfying the hypothesis of Theorem 7.5.2 (ii) is really a strict local max point.*

We next take the case of a function of two real variables, i.e. $f : \mathbf{R}^2 \to \mathbf{R}$ such that f has continuous second order partial derivatives over $\mathbf{R}^2$. The following are again standard results.

Theorem 7.5.3 *(Necessity) Let $(x^*, y^*) \in \mathbf{R}^2$ be a local min or local max point of f over $\mathbf{R}^2$, then*

$$\left.\frac{\partial f}{\partial x}\right|_{(x^*,y^*)} = 0, \qquad \left.\frac{\partial f}{\partial y}\right|_{(x^*,y^*)} = 0. \tag{7.14}$$

To state the sufficient conditions, we introduce the following notations,

$$A = \left.\frac{\partial^2 f}{\partial x^2}\right|_{(x^*,y^*)}, \qquad B = \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{(x^*,y^*)}, \qquad C = \left.\frac{\partial^2 f}{\partial y^2}\right|_{(x^*,y^*)}.$$

Theorem 7.5.4 *(Sufficiency)*

(i) Let $(x^*, y^*) \in \mathbf{R}^2$ satisfy (7.14). Also let $A > 0$ and $AC - B^2 > 0$. Then (x^*, y^*) is a strict local min point of f over $\mathbf{R}^2$.

(ii) Let $(x^*, y^*) \in \mathbf{R}^2$ satisfy (7.14). Also let $A < 0$ and $AC - B^2 > 0$. Then (x^*, y^*) is a strict local max point of f over $\mathbf{R}^2$.

The first thing which we should try to understand here is that conceptually there is no difference between optimizing a function of one variable or a function of two variables. In the case of one variable, the necessary condition says that the tangent at x^* is parallel to x–axis. For the case of two variables it is exactly the same interpretation, i.e. the tangent plane at (x^*, y^*) is parallel to the xy-plane i.e. z =constant.

Let us try to understand the geometric meaning of the sufficient condition now. In particular we consider Theorem 7.5.2. Given that $f''(x^*) > 0$ and f'' is continuous, we get that $f''(x) > 0$ in certain neighborhood of x^*; i.e. the function looks like a parabola $(\cup)$ locally around x^* and therefore we are justified in declaring x^* as a strict local min point. Similarly, $f''(x^*) < 0$ and f'' is continuous, really mean that f looks like an inverted parabola $(\cap)$ locally around x^*, so x^* can be declared as a strict local max point. Thus the conditions of Theorem 7.5.2 essentially check the shape of the function f locally around x^*; to be precise these conditions check if in the neighborhood of x, f is a strictly convex function or a strictly concave function. Exactly the same thing is being done in Theorem 7.5.4 as well. Because if we construct the matrix

$$Q = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$$

then the given conditions of Theorem 7.5.4 are essentially verifying that the matrix Q is positive definite or negative definite, i.e. f is a strictly convex function or a strictly concave function in a neighborhood of the point (x^*, y^*). So even if Theorem 7.5.3 and Theorem 7.5.4 look different, geometrically they are doing the same thing, checking the strict convexity/strict concavity of the objective function in the neighborhood of a stationary point.

If we now consider a general unconstrained optimization problem, Min (or Max) $f(x)$ over $x \in \mathbf{R}^n$, then under appropriate differentiability assumptions, we can state the following theorems on the lines of earlier Theorems 7.5.3 and 7.5.4.

Theorem 7.5.5 *(Necessity) Let $x^* \in \mathbf{R}^n$ be a local min or local max point of f over $\mathbf{R}^n$, then $\nabla f(x^*) = 0$.*

Theorem 7.5.6 *(Sufficiency) Let $x^* \in \mathbf{R}^n$ such that $\nabla f(x^*) = 0$. If f is a strictly convex (respectively strictly concave) function in a neighborhood of x^*, then x^* is a local min (respectively max) point of $f(x)$ over $\mathbf{R}^n$.*

Since, in practice, it may be difficult to check the shape of the function locally, we assume a general shape (convex/concave) over the entire domain $\mathbf{R}^n$. This gives the following theorem.

Theorem 7.5.7 *(Sufficiency) Let $x^* \in \mathbf{R}^n$ such that $\nabla f(x^*) = 0$. Let f be a convex over $\mathbf{R}^n$, then x^* is a global min point of f over $\mathbf{R}^n$. Similarly if f is a concave function over $\mathbf{R}^n$, then x^* is a global max point of f over $\mathbf{R}^n$.*

Constrained Optimization with Equality Constraints

We next construct the constrained optimization problem where all constraints are given in '=' form, i.e. we wish to optimize $f(x)$ over the set S where

$$S = \{\, x \in \mathbf{R}^n : \; g_i(x) = 0, \; i = 1, 2, \ldots, m \,\}.$$

Traditionally, such problems have been solved by the classical *method of Lagrange multipliers*. Here we construct the Lagrange function or the Lagrangian as

$$L(x, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \,, \quad x \in \mathbf{R}^n, \; \lambda \in \mathbf{R}^m \,.$$

Then we have the following theorem.

Theorem 7.5.8 *(Necessity) Let $x^* \in \mathbf{R}^n$ be a local min or local max point of f over the feasible set $S = \{\, x \in \mathbf{R}^n : \; g_i(x) = 0, \; (i = 1, 2, \ldots, m) \,\}$ where $m < n$. Let it be possible to choose a set of m variables x_i for which the Jacobian matrix $J = \left(\frac{\partial g_i}{\partial x_j}\Big|_{x^*} \right)_{m \times m}$ has an inverse. Then there exists a unique set of Lagrange multipliers $\lambda_1^*, \lambda_2^*, \ldots, \lambda_m^*$ such that*

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0. \tag{7.15}$$

Remark 7.5.2 *We need the invertibility of the Jacobian matrix because the proof requires the application of the implicit function theorem.*

Remark 7.5.3 *The conditions (7.15) together with the constraints $g_i(x^*) = 0$ $(i = 0, 1, \ldots, m)$ give a system of $(n+m)$ equations for the determination of $(n+m)$ unknowns $x_1^*, x_2^*, \ldots, x_n^*, \lambda_1^*, \lambda_2^*, \ldots, \lambda_m^*$. These conditions are essentially*

$$\frac{\partial L}{\partial x_j} = 0 \,, \qquad (j = 1, 2, \ldots, n)$$

and

$$\frac{\partial L}{\partial \lambda_i} = 0 \,, \qquad (i = 1, 2, \ldots, m) \,.$$

Also as $\lambda \in \mathbf{R}^m$ is unconstrained, we can take the Lagrangian also as

$$L(x, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x) - \sum_{i=1}^{m} \lambda_i g_i(x), \quad x \in \mathbf{R}^n, \ \lambda \in \mathbf{R}^m.$$

Now let us try to understand the geometrical meaning of Theorem 7.5.8. For this let us assume that $n = 3$ and $m = 2$, i.e. there are two equality constraints $g_1(x) = 0$ and $g_2(x) = 0$, $x \in \mathbf{R}^3$. Clearly these two constraints describe two surfaces in three dimensional space. If there has to be any feasible solution, these two surfaces must intersect forming a curve C. We can find the minimum value of the objective function by examining the surface $f(x)$ =constant=k (say) as the constant k is gradually decreased. The intersection of any such surface $f(x) = k_0$ with the curve C is a set of feasible points where objective value is k_0; and any local minimum on C must occur at a point P where the surface $f(x) = k_{min}$ is tangent to C.

Next, we note that ∇f is normal to the surface $f(x) = k_{min}$, since k_{min} is constant. Thus ∇f must also be normal to the curve C at the point P. The other two gradients ∇g_1 and ∇g_2 are normal to surfaces $g_1(x) = 0$ and $g_2(x) = 0$ respectively, so both of them are also normal to C. So the plane normal to C at P contains all three gradient vectors, and therefore these must be linearly dependent. Thus there exist scalars $\alpha_0, \alpha_1, \alpha_2$ such that $\alpha_0 \nabla f + \alpha_1 \nabla g_1 + \alpha_2 \nabla g_2 = 0$ and $(\alpha_0, \alpha_1, \alpha_2) \neq 0$. But the constraint surfaces $g_1(x) = 0$ and $g_2(x) = 0$ intersect to form a curve, i.e $\alpha_1 \nabla g_1 + \alpha_2 \nabla g_2 = 0$ is possible only for $\alpha_1 = 0 = \alpha_2$. Therefore $\alpha_0 \neq 0$. Taking $\lambda_1 = \frac{\alpha_1}{\alpha_0}$, and $\lambda_2 = \frac{\alpha_2}{\alpha_0}$ we get conditions of the theorem.

Theorem 7.5.9 *(Sufficiency) Let $(x^*, \lambda^*) \in \mathbf{R}^n \times \mathbf{R}^m$ exists such that $\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0$. Let $Z(x^*) = \{z \in \mathbf{R}^n : z^T \nabla g(x^*) = 0\}$, where $g(x) = col\big(g_1(x), g_2(x), \ldots, g_m(x)\big)$. Also let $H_L(x^*, \lambda^*)$ denote the Hessian of the Lagrangian at the point (x^*, λ^*). Furthermore let $z^T H_L(x^*, \lambda^*) z > 0$ for all $z \in Z(x^*)$ with $z \neq 0$, then x^* is a strict local min point of f subject to $g_i(x) = 0$, $(i = 1, 2, \ldots, m)$.*

In a similar manner, if $z^T H_L(x^*, \lambda^*) z < 0$ for all $z \in Z(x^*)$ with $z \neq 0$, then x^* is a strict local max point of f subject to to $g_i(x) = 0$, $(i = 1, 2, \ldots, m)$.

Interpretation of Lagrange Multipliers

Let us consider the nonlinear programming problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) = b_i \quad (i = 1, 2, \ldots, m). \end{aligned} \tag{7.16}$$

Let problem (7.16) have a global min point x^* and the hypothesis of Theorem 7.5.8 hold; so that the associated Lagrange multipliers $\lambda_1^*, \lambda_2^*, \ldots, \lambda_m^*$ do exist. Then it can be proved that

$$\lambda_i^* = \left.\frac{\partial f(x)}{\partial b_i}\right|_{x = x^*} \quad (i = 1, 2, \ldots, m). \tag{7.17}$$

The above equation tells that the Lagrange multiplier λ_i^* gives the rate of change of the optimal attainable value of the objective function $f(x)$ with respect to change in b_i. Thus if the ith resource b_i is changed to $b_i + \Delta b_i$ then the objective function value is expected to change by an amount $\lambda^* \Delta b_i$. We have already read such a result for LPP and the same holds for problems of type (7.16) as well.

Example 7.5.1 *Use method of Lagrange multipliers to solve*

$$\begin{aligned} \textit{Min} \quad & \frac{x^3}{3} - \frac{3y^2}{2} + 2x \\ \textit{subject to} \quad & \\ & x - y = 0. \end{aligned}$$

Solution We have $f(x, y) = \frac{x^3}{3} - \frac{3y^2}{2} + 2x$ and $g(x, y) = x - y$. Therefore the Lagrangian is

$$L(x, y, \lambda) = \frac{x^3}{3} - \frac{3y^2}{2} + 2x + \lambda(x - y).$$

Evaluating the partial derivatives of the Lagrangian $L(x, y, \lambda)$ w.r.t x, y, λ and equating each equal to zero we get

$$\begin{aligned} x^2 + 2 + \lambda &= 0 \\ -3y - \lambda &= 0 \\ x - y &= 0. \end{aligned}$$

The solutions of above system are $(x = 2, y = 2, \lambda = -6)$ and $(x = 1, y = 1, \lambda = -3)$. We next compute the Hessian of the Lagrangian, i.e.

$$\begin{aligned} H_L(x, y, \lambda) &= \begin{bmatrix} \dfrac{\partial^2 L}{\partial x^2} & \dfrac{\partial^2 L}{\partial x \partial y} \\ \dfrac{\partial^2 L}{\partial x \partial y} & \dfrac{\partial^2 L}{\partial y^2} \end{bmatrix} \\ &= \begin{bmatrix} 2x & 0 \\ 0 & -3 \end{bmatrix}. \end{aligned}$$

For $(x^* = 2, y^* = 2)$ we have

$$H_L(x^*, y^*, \lambda^*) = \begin{bmatrix} 4 & 0 \\ 0 & -3 \end{bmatrix}.$$

Also $z^T \nabla g(x^*, y^*) = 0$ means that $(z_1 - z_2) = 0$, where $z = (z_1, z_2)^T$. Then $z^T H_L z = 4z_1^2 - 3z_2^2 = z_1^2 > 0$ for $z \neq 0$. Therefore by Theorem 7.5.9, the point $(2,2)$ is a strict local min point. In a similar manner we can check that $(1,1)$ is a strict local max point. We may also note here that the given problem has neither global min point nor global max point because $f(x,0) \to +\infty$ as $x \to \infty$ and $f(x,0) \to -\infty$ as $x \to -\infty$.

Constrained Optimization with Inequality Constraints

We consider the inequality constrained optimization problem

$$\begin{aligned} &Min && f(x) \\ &\text{subject to} \\ & && g_i(x) \leq b_i\,, \quad (i = 1,2,\ldots,m) \end{aligned} \tag{7.18}$$

and try to get the optimality condition for the same. Here we present a very elementary and non rigorous development of the Karush-Kuhn-Tucker (KKT) optimality conditions which is very much motivated by the method of Lagrange multipliers as discussed above. But we would like to emphasize that the KKT optimality conditions constitute a very important core topic in optimization which is important, both from theoretical as well as algorithmic point of view and on which we shall devote a full chapter later in the book.

Looking at problem (7.18), it is very natural to convert each inequality constraint $g_i(x) \leq b_i$ into an equation by adding a squared slack variable s_i^2 so that the given problem can be rewritten as

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) + s_i^2 = b_i \quad (i = 1,2,\ldots,m)\,. \end{aligned} \tag{7.19}$$

Now problem (7.19) is an optimization problem with equality constraints and so the method of Lagrange multipliers is applicable. Therefore we construct the Lagrangian

$$L(x, s, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i (g_i(x) + s_i^2 - b_i)$$

and use Theorem 7.5.8 to write the necessary conditions at the optimal point (x^*, s^*, λ^*). This gives

$$\nabla_x L = 0 \Rightarrow \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0 \tag{7.20}$$

$$\frac{\partial L}{\partial s_i} = 0 \Rightarrow \lambda_i^* s_i^* = 0 \quad (i = 1,2,\ldots,m) \tag{7.21}$$

$$\frac{\partial L}{\partial \lambda_i} = 0 \Rightarrow g_i(x^*) + s_i^2 - b_i = 0 \quad (i = 1,2,\ldots,m)\,. \tag{7.22}$$

So far $\lambda_i's$ are unconstrained as per the statement of Theorem 7.5.8. However from (7.17) we also know that $\lambda_i^* = \partial f(x^*)/\partial b_i'$ where $b_i' = b_i - s_i^2$ because the constraint $g_i(x) \leq b_i$ has been expressed as $g_i(x) + s_i^2 = b_i$.

Now if we increase the value of b_i (i.e. resource b_i is available in $b_i + \delta b_i$, $\delta b_i > 0$, units) then because the constraints are '$\leq$' type, the feasible region of (7.19) will be enlarged so we have more options now. Therefore the value of the objective function $f(x^*)$ will improve and hence $\lambda_i^* \geq 0$. Also the equation (7.21) can be rewritten as $\lambda_i^*(s_i^*)^2 = 0$, i.e. $\lambda_i^*(g_i(x^*) - b_i) = 0$. Therefore the optimality conditions are

$$\begin{aligned} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) &= 0 \\ \lambda_i^*(g_i(x^*) - b_i) &= 0 \qquad (i = 1, 2, \ldots m) \\ g_i(x^*) &\leq b_i \qquad (i = 1, 2, \ldots m) \\ \lambda_i^* &\geq 0 \qquad (i = 1, 2, \ldots m)\,. \end{aligned}$$

These are precisely the celebrated Karush-Kuhn-Tucker (KKT) conditions for the inequality constrained optimization problem (7.18).

In the above we must note that as we are using Theorem 7.5.8, so the condition of the invertibility of the Jacobian should also hold. This condition for the changed scenario leads to a well known concept of *constraint qualification*. We shall have opportunity to discuss all these things in greater detail in Chapter 8 of the book.

7.6 Quadratic Programming

We introduced the class of convex programming problems in Section 7.4 and developed the KKT necessary/sufficient optimality conditions for the same in Section 7.5. Before developing algorithms for the general convex programming problem, here in this section we restrict ourselves to a very special case of convexity, i.e. the convexity of a positive semi-definite quadratic form in n variables. This leads to a quadratic programming problem (QPP), i.e an optimization problem where we are minimizing (maximizing) a positive (negative) semi-definite quadratic form of n variables subject to linear inequalities or equations. Recalling certain results of earlier sections, we note that the linearity of the constraints guarantees that the feasible region is a convex set and the convexity of the objective function assures that every local min point is also a global min point. Here we may note that, unlike LPP's, it is not true that the optimal solution of a quadratic programming problem has to lie at a corner point or even at the boundary of the feasible region. At this stage, it may not be a bad idea to construct examples of QPP's in two variables to illustrate that, in general, optimal point can be anywhere - be a corner point, a boundary point or an interior point of the feasible region.

There are two main algorithms for solving (QPP). These are *Wolfe's method* and *Beale's method*. While Wolfe's method requires that the matrix of the quadratic form to be minimized is positive definite, Beale's method is applicable even when the matrix is positive semi-definite. In our presentation here, we shall discuss Wolfe's method only, mainly because of two reasons. Firstly, this algorithm gives an immediate application of the KKT conditions in the algorithmic development of nonlinear programming problems, and secondly it uses only the Phase-I of the simplex algorithm with certain appropriate modifications. At the first reading, we may wonder how the simplex algorithm enters here because that is used to solve LPP's, but if we think for a moment we get the answer immediately. As the objective function in QPP is a positive definite quadratic form and the constraints are linear, the KKT conditions will 'almost' be a system of linear equations in non-negative variables. We have already seen that the Phase-I of the simplex algorithm could be employed to solve a linear system for non-negative variables and therefore it is natural to employ the same to solve QPP's.

7.7 Wolfe's Method for Quadratic Programming

We consider the following quadratic programming problem

$$\begin{aligned} &\text{Max} && c^T x + x^T D x \\ &\text{subject to} \\ & && Ax \leq b \\ & && x \geq 0\,, \end{aligned} \tag{7.23}$$

where $c \in \mathbf{R}^n, x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $A = [a_{ij}]$ is an $(m \times n)$ matrix, and $D = [d_{ij}]$ is an $(n \times n)$ negative semi-definite matrix.

Here we note that D is taken to be negative semi-definite because problem (7.23) is in the maximization form. This makes the objective function a concave function as required to be for the maximization case.

We now write the KKT conditions for problem (7.23). For this we write problem (7.23) in the form in which the KKT conditions have been stated in the last section. This leads to

$$\begin{aligned} &\text{- Min} && (-f(x)) \\ &\text{subject to} \\ & && G(x) \leq 0 \\ & && -Ix \leq 0\,, \end{aligned} \tag{7.24}$$

where $-f(x) = -c^T x - x^T D x$, $G(x) = Ax - b$ and I is an identity matrix of the appropriate order.

Since in the above problem, the minus sign is outside the minimization, we need to solve the problem

$$\begin{aligned} &\text{Min} && (-f(x)) \\ &\text{subject to} \\ & && G(x) \le 0 \\ & && -Ix \le 0, \end{aligned} \tag{7.25}$$

with f and G as defined before. Then problems (7.23) and (7.25) will have the same optimal solutions, though the optimal value of the given problem (7.23) will be the negative of the optimal value of problem (7.25).

Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ and $\mu = (\mu_1, \mu_2, \ldots, \mu_m)^T$ be the Lagrange multipliers corresponding to the constraints $Ax - b \le 0$ and $-x \le 0$ respectively. Then

$$\nabla(-f(x)) = -\nabla f(x) = -c - 2Dx, \ \nabla G(x) = A \text{ and } \nabla(-Ix) = -I.$$

Here $\nabla G(x)$ is to be understood as $[\nabla g_1(x), \ldots, \nabla g_m(x)]^T$, $g_i(x)$ being $(\sum_{j=1}^{n} a_{ij}x_j - b_i)$ for $i = 1, 2, \ldots, m$. The expression $\nabla(-Ix) = -I$ is understood similarly for the constraint $-x \le 0$. Hence the KKT conditions for problem (7.25) are

$$\begin{aligned} -c - 2Dx + A^T\lambda - I\mu &= 0 \\ \lambda_i\Big(\sum_{j=1}^{n} a_{ij}\, x_j - b_i\Big) &= 0 \quad (i = 1, 2, \ldots, m) \\ \mu_j x_j &= 0 \quad (j = 1, 2, \ldots, n) \\ Ax - b &\le 0 \\ x, \lambda, \mu &\ge 0. \end{aligned} \tag{7.26}$$

Now defining $s_i = b_i - \sum_{j=1}^{n} a_{ij}\, x_j$, $(i = 1, 2, \ldots, m)$ and $s = (s_1, s_2, \ldots, s_m)^T$, we have from (7.26)

$$\begin{aligned} -2Dx + A^T\lambda - I\mu &= c \\ Ax + s &= b \\ x, \lambda, \mu, s &\ge 0 \\ \lambda_i s_i &= 0 \quad (i = 1, 2, \ldots, m) \\ \mu_j x_j &= 0 \quad (j = 1, 2, \ldots, n)\,. \end{aligned}$$

We can further rewrite the above KKT system in the following matrix form

$$\begin{bmatrix} -2D & A^T & -I & 0 \\ A & 0 & 0 & I \end{bmatrix} \begin{pmatrix} x \\ \lambda \\ \mu \\ s \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}$$

$$\begin{aligned} x, \lambda, \mu, s &\geq 0 \\ \lambda_i s_i &= 0 \quad (i = 1, 2, ..., m) \\ \mu_j x_j &= 0 \quad (j = 1, 2, ..., n). \end{aligned} \tag{7.27}$$

Now we recall the sufficient part of KKT Theorem and infer that if $(x^*, \lambda^*, \mu^*, s^*)$ is a solution of the above KKT system (7.27) then x^* is a global min point of problem (7.25) or equivalently x^* is a global max point of problem (7.23). So now the main goal should be to solve the given KKT system (7.27) efficiently. For this we note that the KKT system consists of two structurally different subsystems. The first subsystem is essentially a system of linear equations in non-negative variables x, λ, μ and s, which can always be solved by employing the Phase-I of the simplex algorithm. The second subsystem consists of is the complementary slackness conditions $\lambda_i s_i = 0 = \mu_j x_j$ for all i and j, which defines a system of nonlinear equations. Though the first system is easy to solve (by using the Phase-I of the simplex algorithm) its solution need not satisfy the complementary slackness conditions, namely, $\lambda_i s_i = 0 = \mu_j x_j$ for all i and j. However looking at these conditions carefully, we note that these conditions imply that

$$\lambda_i > 0 \Longrightarrow s_i = 0 \ (s_i > 0 \Longrightarrow \lambda_i = 0),$$

and

$$\mu_j > 0 \Longrightarrow x_j = 0 \ (x_j > 0 \Longrightarrow \mu_j = 0) \ .$$

Therefore, in the absence of degeneracy, the complementary slackness conditions imply that while solving the first subsystem by the Phase-I of the simplex algorithm we should not make λ_i and s_i as basic variables at the same time, and similarly we should not make μ_j and x_j as basic variables at the same time. In other words, we should make λ_i as an entering variable at the current iteration only when we are sure that s_i will be a non-basic variable in the next tableau. Similar arguments hold for other variables μ_j and x_j as well. Thus in any simplex tableau only one of λ_i and s_i (for the same i) and only one of μ_j and x_j (for the same j) should be a basic variable.

In view of the above, we should explore if such a *restricted entry simplex algorithm* is possible. This is called 'restricted entry' because of the restriction that only one of λ_i and s_i (for a given i) and only one of μ_j and x_j (for a given j) can be a basic variable at a given time. Fortunately Wolfe's algorithm demonstrates that such a procedure, namely the 'restricted entry simplex algorithm' is always possible, provided the matrix D is negative definite.

We now have the following theorem.

Theorem 7.7.1 *If D is negative definite then the quadratic programming problem (7.23) can not have an unbounded solution.*

Proof. Take any finite r (> 0) and consider any point x lying on the hyper sphere $\|x\| = r$. Then $x = r\,d$ where d is a point on the unit hyper sphere with center as origin. Now

$$\begin{aligned} x^T Dx &= r^2 d^T Dd \\ &\le r^2 \underset{\|d\|=1}{Max}\, d^T Dd \\ &= r^2 d_0^T Dd_0 \text{ (say)} \\ &= r^2 m_0 \text{ (say)} \\ &< 0\,. \end{aligned} \tag{7.28}$$

Here it may be noted that m_0 definitely exists because $d^T Dd$ is a continuous function of d and $\{d : \|d\| = 1\}$ is a closed bounded set of $\mathbf{R}^n$. Further $d_0 \neq 0$ and so by the negative definiteness of D, m_0 is less than zero. Thus, as $\|x\| \to +\infty$, $x^T Dx \to -\infty$.

Further, for a non-zero x, we have

$$\begin{aligned} f(x) &= c^T x + x^T Dx \\ &= x^T Dx\Big[1 + \frac{c^T x}{x^T Dx}\Big] \end{aligned}$$

But

$$\begin{aligned} \left|\frac{c^T x}{x^T Dx}\right| &= \frac{1}{r}\left|\frac{c^T d}{d^T Dd}\right| \\ &\le \frac{1}{r}\underset{|d|=1}{Max}\left|\frac{c^T e}{d^T Dd}\right| \\ &= \frac{1}{r}\left|\frac{c^T d_1}{d_1^T Dd_1}\right| = \frac{m_1}{r} \text{ (say).} \end{aligned}$$

Here d_1 is the point at which the maximum of $\left|\frac{c^T d}{d^T Dd}\right|$ is attained and m_1 is the maximum value.

Now, above tells that, $\frac{c^T x}{x^T Dx} \to 0$ as $\|x\| \to \infty$. Hence $f(x) \to -\infty$ as $\|x\| \to +\infty$.

Thus what essentially we have proved above is that the maximum of $(c^T x + x^T Dx)$ over $\mathbf{R}^n$ can not be unbounded. The result of Theorem 7.7.1 follows immediately from the fact that the set of feasible solutions of problem (7.23) is a subset of $\mathbf{R}^n$. □

Here, it must be noted that Theorem 7.7.1 may not hold if D is negative semi-definite. For this we have the following example

Example 7.7.1 *Use Wolfe's method to solve the following QPP*

$$\begin{aligned} &\textit{Max} \quad z = x_1 + x_2 - (2x_1 - x_2)^2 \\ &\textit{subject to} \\ &\qquad x_1 - x_2 \le 1 \\ &\qquad x_1,\ x_2 \ge 0. \end{aligned}$$

Solution We have $x = (x_1, x_2)^T, c = (1,2)^T, b = (1), A = [1, -1]$ and

$$D = \begin{pmatrix} -4 & 2 \\ 2 & -1 \end{pmatrix}.$$

Obviously D is negative semi-definite. Now observe that for any $\theta \ge 0$,

$$x_1 = \theta, x_2 = 2\theta$$

is a feasible solution of the problem and the value of the objective function is

$$z = \theta + 2\theta - (2\theta - 2\theta)^2 = 3\theta\ .$$

Thus as $\theta \to \infty$, $z \to \infty$. So the given problem has an unbounded solution.

The above example suggests that possibly Theorem 7.7.1 is true even for the negative semi-definite case, provided $c = 0$. This is correct and readers may like to verify the same.

In view of Theorem 7.7.1, some finite feasible point x^* must be a global maximum of problem (7.23). Hence by the necessary part of the KKT theorem, it is necessary that x^* satisfies the corresponding KKT system. Thus the above KKT system definitely has a solution.

Remark 7.7.1 *In case D is negative definite, the objective function of the quadratic programming problem (7.23) is strictly concave and so if it has a feasible solution then it has unique optimal solution. However when D is negative semi-definite then to guarantee that it has bounded optimal solution we not only need that the given problem (7.23) is feasible but also require that $c = 0$ (refer to the proof of Theorem 7.7.1 above). Therefore we expect that Wolfe's method will converge for the case (i) D is negative definite or (ii) D is negative semi-definite with $c = 0$, and this is really true.*

We now describe Wolfe's algorithm to find a solution of the KKT system.

Wolfe's Method: A Stepwise Description

The stepwise description of Wolfe's method for solving the quadratic programming problem (7.23) is now described below. Here we assume that the given QPP has a feasible solution; a fact that can always be verified by using the Phase-I of the usual simplex method.

Step 1 Ignore the complementary slackness conditions in the KKT system (7.27) and consider the remaining system of linear equations in the non-negative variables x, λ, μ and s.

Step 2 Check that in the linear system, all components of c and b are non-negative, If some component of c or b is negative, multiply the corresponding equation by -1.

Step 3 After performing Step 2, add appropriate number of artificial variables x_{a_i} to get an identity matrix of order $(m+n)$. Construct the Phase-I problem with the usual objective function as $-\sum_k x_{a_k}$. At this stage we should note that for the initial b.f.s of the Phase-I problem so constructed, the desired complementary slackness conditions $(\lambda_i s_i = \beta_j x_j = 0)$ for $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ can be assumed to hold automatically. This is because, if need be, we can add artificial variables to all constraints, even though an identity column is present, so that the initial b.f.s consist of artificial variables only.

Step 4 Solve the problem in Step 3 by the restricted basis entry method. (This method is the same as the usual simplex method, except that while entering a column, choose the one for which the relative cost $z_j - c_j$ is negative and which does not make both λ_i and s_i or μ_j and x_j as basic variables at the same time.)

step 5 Stop when either all relative costs are non-negative or it is not possible to enter a non basic column with a negative reduced cost, without violating the complementary slackness conditions. The basic solution so obtained will be the optimal solution of the given quadratic programming problem (see the convergence theorem, Theorem 7.7.2).

The above step will be justified provided we prove that at the end of Step 5, all artificial variables will be at the zero level. Before we prove this, we wish to illustrate the algorithm with the help of following examples.

Example 7.7.2 *Use Wolfe's method to solve the following QPP*

$$\text{Max} \quad z = x_1 + x_2 - x_1^2 + 2x_1x_2 - 2x_2^2$$

subject to

$$2x_1 + x_2 \le 1$$
$$x_1,\ x_2 \ge 0.$$

Solution Here $x = (x_1, x_2)^T, c = (1,1)^T, b = (1), A = [2,1]$ and $D = \begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix}$

Clearly D is negative definite. Now, noting that λ and s will have only one component each and μ will have two components, we have the following KKT system for the given QPP

$$\begin{pmatrix} 2 & -2 & 2 & -1 & 0 & 0 \\ -2 & 4 & 1 & 0 & -1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \mu_1 \\ \mu_2 \\ s_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{7.29}$$

$$x_1, x_2, \lambda_1, \mu_1, \mu_2, s_1 \geq 0,$$

and

$$\lambda_1 s_1 = \mu_1 x_1 = \mu_2 x_2 = 0.$$

Since the slack variable s_1 has already given one identity column, and all the components of c, and b are non-negative, we add only two artificial variables, namely x_{a_1} and x_{a_2} and get the following Phase-I problem

Max $\quad -x_{a_1} - x_{a_2}$
subject to

$$\begin{pmatrix} 2 & -2 & 2 & -1 & 0 & 1 & 0 & 0 \\ -2 & 4 & 1 & 0 & -1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \mu_1 \\ \mu_2 \\ x_{a_1} \\ x_{a_2} \\ s_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{7.30}$$

$$x_1, x_2, \lambda_1, \mu_1, \mu_2, x_{a_1}, x_{a_2}, s_1 \geq 0,$$

and

$$\lambda_1 s_1 = \mu_1 x_1 = \mu_2 x_2 = 0.$$

The starting tableau for the above is

	x_1	x_2	λ_1	μ_1	μ_2	x_{a1}	x_{a_2}	s_1
$x_{a_1} = 1$	2	-2	2	-1	0	1	0	0
$x_{a_2} = 1$	-2	$\boxed{4}$	1	0	-1	0	1	0
$s_1 = 1$	2	1	0	0	0	0	0	1
-2	0	-2	-3	1	1	0	0	0

Now there is something important to note. Had it been the usual simplex method, we would have made λ_1 a basic variable and x_{a_1}, a non-basic variable in the next iterations. But here we have also to take care of the constraints

$$\lambda_1 s_1 = \mu_1 x_1 = \mu_2 x_2 = 0.$$

Observe that initially these constraints are automatically satisfied because $\lambda_1, \mu_1, \mu_2, x_1$ and x_2 are non-basic variables. However, if λ_1 is made a basic variable then s_1 can not be made a non basic variable (because the corresponding y_{ij} entry is zero) and so this will make both λ_1 and s_1 as basic variables, something which we don't want.

Hence in the restricted basis entry simplex method, the column to enter the basis will be of x_2. Then by the usual criterion, x_{a_2} will go out of basis. This gives the following tableau

	x_1	x_2	λ_1	μ_1	μ_2	x_{a_1}	x_{a_2}	s_1
$x_{a_1} = 3/2$	1	0	5/2	-1	-1/2	1	1/2	0
$x_2 = 1/4$	-1/2	1	1/4	0	-1/4	0	1/4	0
$s_1 = 3/4$	5/2	0	-1/4	0	1/4	0	-1/4	1
-3/2	-1	0	-5/2	1	1/2	0	1/2	0

Again we can not enter λ_1 as s_1 can not go out because (the corresponding $\hat{y}_{ij}$ entry is negative). Thus in the next iteration, x_1 will become a basic variable and s_1 a non basic variable. The new tableau is

	x_1	x_2	λ_1	μ_1	μ_2	x_{a_1}	x_{a_2}	s_1
$x_{a_1} = 6/5$	0	0	13/5	-1	-3/5	1	1	-2/5
$x_2 = 2/5$	0	1	1/5	0	-1/5	0	0	1/5
$x_1 = 3/10$	1	0	-1/10	0	1/10	0	-1/10	2/5
-6/5	0	0	-13/5	1	3/5	0	0	2/5

Next λ_1 will become a basic variable and it will not violate the condition $\lambda_1 s_1 = 0$ because s_1 is already a non basic variable. The variable to become non basic will be x_{a_1}. Thus we have the following tableau

	x_1	x_2	λ_1	μ_1	μ_2	x_{a_1}	x_{a_2}	s_1
$\lambda_1 = 6/13$	0	0	1	-5/13	-3/13	5/13	5/13	-2/13
$x_2 = 4/13$	0	1	0	1/13	-2/13	-1/13	-1/13	3/13
$x_1 = 9/26$	1	0	0	-1/26	1/13	1/26	-8/130	5/13
0	0	0	0	0	0	1	1	0

Now $(x_1^* = 9/26, x_2^* = 4/13, \lambda_1^* = 6/13, \mu_1^* = 0, \mu_2^* = 0, s_1^* = 0)$ is a solution of problem (7.30) and hence a solution of the KKT system (7.29) as well. Therefore from the sufficiency part of the KKT theorem the optimal solution to the given QPP is $(x_1^* = 9/26, x_2^* = 4/13)$.

Example 7.7.3 *Use Wolfe's method to solve the following QPP.*

Max $z = 2x_1 + 3x_2 - 3x_1^2 + 2x_1x_2 - 2x_2^2$

subject to

$$\begin{aligned} x_1 + x_2 &\geq 1 \\ 3x_1 + 4x_2 &\leq 12 \\ x_1,\ x_2 &\geq 0. \end{aligned}$$

Solution First, the above problem is expressed in the form in which Wolfe's method is applicable, i.e,

Max $z = 2x_1 + 3x_2 - 3x_1^2 + 2x_1x_2 - 2x_2^2$
subject to

$$-x_1 - x_2 \le -1$$
$$3x_1 + 4x_2 \le 12$$
$$x_1,\ x_2 \ge 0.$$

Now $c = (2,3)^T, x = (x_1, x_2)^T, b = (-1, 12)^T$ and

$$A = \begin{pmatrix} -1 & -1 \\ 3 & 4 \end{pmatrix}, \quad D = \begin{pmatrix} -3 & 1 \\ 1 & -2 \end{pmatrix}$$

We can also check that D is negative definite and the given QPP can be solved by Wolfe's method. The KKT system to be solved is

$$\begin{pmatrix} 6 & -2 & -1 & 3 & -1 & 0 & 0 & 0 \\ -2 & 4 & -1 & 4 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \\ \mu_1 \\ \mu_2 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ -1 \\ 12 \end{pmatrix}$$

$$x_j, \lambda_i, \mu_j, s_i \ge 0 \quad (i = 1,2; j = 1,2)$$
$$\lambda_i s_i = 0 \quad (i = 1,2)$$

and

$$\mu_j x_j = 0 \quad (j = 1,2).$$

Since the third component in the R.H.S is negative, we multiply the third equation by -1 and then add three artificial variables, namely x_{a_1}, x_{a_2}, and x_{a_3} to get the following Phase-I problem:

Max $-x_{a_1} - x_{a_2} - x_{a_3}$
subject to

$$\begin{pmatrix} 6 & -2 & -1 & 3 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -2 & 4 & -1 & 4 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \mu \\ s_1 \\ x_{a_1} \\ x_{a_2} \\ x_{a_3} \\ s_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 12 \end{pmatrix}$$

$$x_j, \lambda_i, \mu_j, s_i \geq 0 \quad (i = 1,2; j = 1,2)$$

$$\lambda_i s_i = 0 \quad (i = 1,2)$$

and

$$\mu_j x_j = 0 \quad (j = 1,2).$$

Here $x = (x_1, x_2)^T$, $\lambda = (\lambda_1, \lambda_2)^T$ and $\mu = (\mu_1, \mu_2)^T$. The problem can now be solved by the restricted entry simplex method in a manner similar to Example (7.7.2). We can verify that $(x_1^* = 7/10, x_2^* = 11/10)$, is the optimal solution of the given QPP and the optimal value is 47/20.

Theorem 7.7.2 (Convergence Theorem).
For the quadratic programming problem (7.23), let either (i) D is negative definite or (ii) D is negative semi-definite and c = 0. Then, in the absence of degeneracy, Wolfe's method always converges in finite number of iterations.

Here 'convergence' has to be understood in the sense that at the end of Step 5 of Wolfe's method, all artificial variables x_{a_i} will be zero and the method will produce KKT point in finite number of iterations. Although we shall not prove this result, we shall verify the same through examples only.

For the case when D is negative definite we have already seen the convergence in Examples 7.7.2, and 7.7.3. So now we should take examples where D is negative semi-definite and consider two cases, namely $c = 0$ and $c \neq 0$. For the example in the first case, the algorithm should converge but that may not happen for the second case.

Example 7.7.4 *Use Wolfe's method to solve the following (QPP).*

$$\begin{aligned} &\textit{Max} \quad z = -x_1^2 + 2x_1x_2 - x_2^2 \\ &\textit{subject to} \\ &\qquad 2x_1 + x_2 \leq 1 \\ &\qquad x_1,\ x_2 \geq 0. \end{aligned}$$

Solution Here, $c = (0,0)^T, A = [2 \quad 1], b = (1)$ and $D = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$, which is negative semi-definite. As $c = 0$, so we expect the algorithm to converge.

For the given problem the KKT conditions are:

$$\begin{pmatrix} 2 & -2 & 2 & -1 & 0 & 0 \\ -2 & 2 & 1 & 0 & -1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \mu_1 \\ \mu_2 \\ s_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$x_1, x_2, \lambda_1, \mu_1, \mu_2, s_1 \geq 0,$$
$$\lambda_1 s_1 = \mu_1 x_1 = \mu_2 x_2 = 0.$$

Now the first restricted entry simplex tableau is

	x_1	x_2	λ_1	μ_1	μ_2	x_{a_1}	x_{a_2}	s_1
$x_{a_1} = 0$	2	-2	2	-1	0	1	0	0
$x_{a_2} = 0$	-2	2	1	0	-1	0	1	0
$s_1 = 1$	2	1	0	0	0	0	0	1
0	0	0	-3	0	0	0	0	0

Here if we have to enter λ_1 (as it is the only variable for which cost coefficient is negative) then in the next tableau λ_1 and s_1 both will become basic variables (note that s_1 can not leave the basis because the corresponding y_{ij} value is zero). So as per the convergence theorem, all x_{a_i} should be zero and that is happening here. Therefore an optimal solution of the given QPP is $x_1^* = 0$, $x_2^* = 0$ as $(x_1^* = 0, x_2^* = 0, \lambda_1^* = 0, \mu_1^* = 0, \mu_2^* = 0, s_1^* = 1)$ is a solution of KKT system.

The convergence essentially follows because if we force to enter λ_1 then it is entering here at a 'zero' level, i.e in the next tableau, though it will be a basic variable but its value will be zero. Thus no complementary slackness conditions will be violated. The next example suggests that this may not happen if D is negative semi-definite and $c \neq 0$.

Example 7.7.5 *Use Wolfe's method to solve the following (QPP).*

$$\begin{aligned} &\textit{Max} \quad z = 2x_1 + x_2 - x_1^2 + 2x_1x_2 - x_2^2 \\ &\textit{subject to} \\ &\qquad 2x_1 + x_2 \leq 1 \\ &\qquad x_1,\ x_2 \geq 0. \end{aligned}$$

Solution Here, $c = (2,1)^T, A = [2 \quad 1], b = (1)$ and $D = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$. The matrix D is negative semi-definite as the corresponding quadratic form equals $-(x_1 - x_2)^2$ which is less than or equal to zero for $x \in \mathbf{R}^2$ and it is zero for $x_1 = x_2 \neq 0$. Therefore the initial tableau of Wolfe's method is

	x_1	x_2	λ_1	μ_1	μ_2	x_{a_1}	x_{a_2}	s_1
$x_{a_1} = 2$	2	-2	2	-1	0	1	0	0
$x_{a_2} = 1$	-2	2	1	0	-1	0	1	0
$s_1 = 1$	2	1	0	0	0	0	0	1
-3	0	0	-3	1	1	0	0	0

Now only one $(z_j - c_j)$ is negative and it corresponds to the variable λ_1. But λ_1 cannot be entered in the basis as $s_1 (> 0)$ is already a basic variable and it cannot leave the basis because the corresponding $y_{ij} = 0$. Therefore there is no way to proceed with

the algorithm and it fails to give an optimal solution of the given QPP. In this context it may be noted that the given QPP certainly has an optimal solution as the feasible region is a polytope and the objective function is a continuous function of x_1 and x_2.

7.8 Summary and Additional Notes

- This chapter presents an elementary introduction of the convex/cancave functions and the convex optimization problem.
- Section 7.2 discusses various characterizations and properties of the convex function which are relevant to the study of finite dimensional optimization problems.
- Sections 7.4 and 7.5 are devoted to the study of convex programming problems. The main results presented here are the KKT necessary/sufficient optimality conditions which are motivated by the well known theorems of calculus for minimizing/maximizing a function of n variables.
- This chapter also discusses the quadratic programming problem (QPP) where we maximize a concave quadratic form subject to linear constraints.
- The main method for solving QPP's, namely Wolfe's method, is presented in Section 7.7. It is shown that this method is convergent of either (1) D is negative definite or D is negative semidefinite and $c = 0$.
- Jensen is generally credited for introducing convex functions in 1905, though Hadamard in 1893 and Hölder in 1889 also had related work on this topic.
- Some of the standard references on convex sets and convex functions are Rockafellar [136]. Roberts and Varberg [131], Stoer and Witzgall [150] and Borwein and Lewis [26]. Most of the standard texts on nonlinear programming also have chapters on convex functions, e.g. Mangasarian [109], Bazaraa and Shetty [11] and Avriel [6]. The book by Boyd and Vandenberghe [27] is a recent addition to the literature which gives a complete modern theory and algorithmic development of the convex optimization problems.
- Wolfe's method is directly based on the KKT system of the given QPP and is solved by the restricted entry simplex method. This method was developed by P. Wolfe in 1959.
- Another popular method for solving QPP's is the Beale's method, developed by E. M. L. Beale in 1959.
- Though the topic of quadratic programming is included in most of texts on nonlinear programming , e.g., Avriel [6], Bazarra and Shetty [11] and Simmons [143], there are certain dedicated books on quadratic programming itself e.g. Boot [23] and Van de Panne [121].
- Certain structured QPP's have become very important in the recent past because they are potentially very useful in the areas of *machine learning* and *portfolio optimization*. In particular, if the given QPP has only one linear constraint or has only

box constraints (such problems occur very naturally in the area of machine learning) then we can use some of the recently developed and very efficient algorithms due to Dai and Fletcher [43, 58].

- Quadratic programming is very closely related to the linear complementary problem and bi-matrix games. Infact, we can also solve QPP's by employing Lemke's complementary pivoting algorithm developed by Lemke in 1962.

7.9 Exercises

7.1 *Test whether each of the following functions is convex, concave, or neither convex nor concave*

1. $f(x) = 6x - x^2 - 3,\ x \in \mathbf{R}$.
2. $f(x_1, x_2) = x_1x_2 + x_1 + x_2,\ (x_1, x_2) \in \mathbf{R}^2$.
3. $f(x_1, x_2) = x_1^2 + 3x_1x_2 + 2x_2^2 - 10x_1 - 10x_2,\ (x_1, x_2) \in \mathbf{R}^2$.
4. $f(x_1, x_2) = x_1^2 + 2x_1x_2 + 2x_2^2 - 5x_1 + 4x_2,\ (x_1, x_2) \in \mathbf{R}^2$.
5. $f(x_1, x_2) = -x_1^2 + 2x_1x_2 + 3x_1x_3 + 6x_2x_3,\ (x_1, x_2, x_3) \in \mathbf{R}^3$.

7.2 *Let $f : \mathbf{R}^2 \to \mathbf{R}$ be given by $f(x_1, x_2) = ax_1^2 + bx_2^2 + 2cx_1x_2 + d$. Find values of a, b, c and d for which (i) f is convex (ii) f is concave.*

7.3 *Let $f : \mathbf{R} \to \mathbf{R}$ be a convex function. Show that $g(y) = f(2 - y)$, $y \in \mathbf{R}$, is a concave function on $\mathbf{R}$.*

7.4 *Let $f : \mathbf{R} \to \mathbf{R}$ be a convex function. Let $g(x) = f(f(x))$. Under what conditions, is the function g convex? Verify your answer for (i) $f(x) = x^2$ (ii) $f(x) = e^{-x}$ and (iii) $f(x) = (1 - x)$.*

7.5 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : \mathbf{R} \to \mathbf{R}$. Show that f is a convex function if and only if for any integer $k \geq 2$, we have $f\left(\sum_{i=1}^{k} \lambda_i x^{(i)}\right) \leq \sum_{i=1}^{n} \lambda_i f(x^{(i)})$, for all $x^{(1)}, \ldots, x^{(k)} \in S$ and for all $0 \leq \lambda \leq 1$ $(i = 1, 2, \ldots, k)$ with $\sum_{i=1}^{k} \lambda_i = 1$.*

7.6 *Let $f(x_1, x_2, \ldots, x_n) = \ln(x_1^{\alpha} x_2^{\alpha} \ldots x_n^{\alpha})$ where $\alpha > 0$ and $x_i \geq 0$ $(i = 1, 2, \ldots, n)$. Is f a concave function?*

7.7 *Is the function $f(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \ln p_i$, $p_i > 0$ $(i = 1, 2, \ldots, n)$, a convex function? Give reason for your answer. (This function is the well known entropy function).*

7.8 *Let $f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i)$ where $x_i \in \mathbf{R}$ for $i = 1, 2, \ldots, m$ (such functions are called separable functions). Show that if each f_i is a convex function of x_i $(i = 1, 2, \ldots, n)$ then f is also a convex function of $x \in \mathbf{R}^n$. Is the converse also true? Give reason for your answer.*

7.9 *Let* $f : [0,2] \to \mathbf{R}$ *be given by* $f(x) = \begin{cases} x, & x \in [0,1] \\ 2-x, & x \in (1,2]. \end{cases}$
Show that f *is a concave function. Let* $g : [0,2] \to \mathbf{R}$ *be a given by* $g(x) = min\ (f(x), x^2)$. *Is* g *also a concave function?*

7.10 *Let* $S \subseteq \mathbf{R}^n$ *be a convex set and* $f : S \to \mathbf{R}$ *be continuous. Then* f *is said to be a midpoint convex function if for all* $x^{(1)}, x^{(2)} \in S$, *we have* $f(\frac{x^{(1)}+x^{(2)}}{2}) \leq \frac{1}{2}(f(x^{(1)}) + f(x^{(2)}))$. *Show that* f *is midpoint convex function if and only if it is a convex function. (Note that the continuity of* f *is crucial here).*

7.11 *Using the definition of convexity, drive the famous arithmetic geometric mean inequality* $\sum_{i=1}^{n} \lambda_i a_i \geq \prod_{i=1}^{n} (a_i)^{\lambda_i}$, *where* a_i *are the given positive numbers and* λ_i *are arbitrary non-negative weights satisfying* $\sum_{i=1}^{n} \lambda_i = 1$.

7.12 *Are the following convex programming problems? Give reasons for your answer*

(1) Max $\ln(1 + x_1) + x_2$
subject to,
$$2x_1 + x_2 \leq 3$$
$$x_1, x_2 \geq 0.$$

(2) Max $x_1 - 2x_2$
subject to,
$$Max(0, x_1) - x_2 \leq 0$$
$$x_1^2 + x_2^2 \leq 4\ .$$

(3) Min $x_1^2 + x_2$
subject to,
$$|x_1| + |x_2| \leq 2$$
$$x_1^2 - x_2 \geq 0.$$

(4) Min $x_1 + x_2$
subject to,
$$-x_1 x_2 + 1 \leq 0$$
$$x_1, x_2 \geq 0.$$

(5) Min $|x - 1| + |x - 4|$
subject to,
$$0 \leq x \leq 5.$$

(6) *Min* $\phi(x)$
subject to,
$0 \le x \le 5$
where $\phi(x) = Max(x^2, (x-2)^2, 4)$.

(7) *Max* $4x_1 + 3x_2$
subject to,
$x_1^2 + x_2^2 \le 1$
or
$(x_1 - 1)^2 + x_2^2 \le 1$.

7.13 *Let* $f : [-1, 3] \to \mathbf{R}$ *be given by*

$$f(x) = \begin{cases} -|x|, & -1 \le x \le 1 \\ -|x-2|, & 1 \le x \le 3. \end{cases}$$

Sketch f, E_f, G_f *and* Γ_α *for* $\alpha = 1$, $\alpha = 0$, *and* $\alpha = -1/2$.

7.14 *Let* $f(x) = x^T Q x$ *be a quadratic form in two variables with* $x = (x_1, x_2)^T$ *and*

$$Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}$$

Let $q_{11} > 0$ *and* $q_{11}q_{22} - q_{12}^2 > 0$. *Show that the given quadratic form is positive definite and hence* f *is a strictly convex function. What can you say about the nature of the quadratic form and the function* f *if* $q_{11} \ge 0$ *and* $q_{11}q_{22} - q_{12}^2 \ge 0$? *Give reasons for your answer. Extend the result for the quadratic forms of* n *variables.*

7.15 *Use the definition of convexity to show that if* X *is a random variable such that* $X \in$ *domain of* f *with probability 1, and* f *is convex, then* $f(E(X)) \le E(f(X))$, *provided the expectation exist (this inequality is the famous Jensen's inequality).*

7.16 *Let* $OABC$ *be a square plate of side 4 meteres. Taking* O *as the origin, let* $P(x, y)$ *be any point on the plate. Let the temperature at the point* $P(x, y)$ *be given by*

$$T(P) = 2xy - 2x^2 - 2y^2 .$$

It is desired to find the hottest point on the plate. Formulate the above as an optimization problem and solve the same to get the desired point.

7.17 *A shopping plaza is triangular in shape with vertices as* $A : (0, 0)$, $B : (1, 0)$ *and* $C : (1, 1)$. *An ATM is to be installed at a point* P: (x, y) *in the plaza such that the sum of squares of its distance from the three corners* A, B *and* C *is least.*

1. *Formulate the above as an optimization problem.*
2. *It is claimed that the above optimization problem can be solved by Wolfe's method. Justify this claim.*
3. *Determine the point P by employing a suitable numerical optimization technique.*

7.18 *Consider the following QPP*

$$\begin{aligned} &\textit{Min} && (x_1 - x_2)^2 + x_2 \\ &\textit{subject to} \\ & && -x_1 + x_2 \leq 0 \\ & && x_1 + 2x_2 \leq 3 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

1. *Express the objective in the standard QPP form $c^T x + x^T D x$. Is D positive definite?*
2. *Solve the above QPP by Wolfe's method and identify difficulties if you face any.*

7.19 *Use Wolfe's method to find a point $P : (x, y)$ in the co-ordinate plane which lies on the line $x + 2y = 4$ and is nearest to the origin.*

7.20 *For a QPP in the form 'Max $c^T x + x^T D x$ subject to $Ax \leq b, x \geq 0$', we have the following KKT conditions.*

$$\begin{bmatrix} 6 & -2 & -1 & 3 & -1 & 0 & 0 & 0 \\ -2 & 4 & -1 & 4 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ u_1 \\ u_2 \\ v_1 \\ v_2 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ -1 \\ 12 \end{bmatrix}$$

$$x_1, x_2, u_1, u_2, v_1, v_2, w_1, w_2 \geq 0,$$

$$v_1 x_1 = v_2 x_2 = u_1 w_1 = w_2 u_2 = 0 .$$

1. *Write the (QPP) being solved.*
2. *Perform one complete iteration of Wolfe's Method to solve the (QPP) as obtained at (1) above.*

7.21 *Let $S = \{(x_1, x_2) \in \mathbf{R}^2 : |x_1| + |x_2| \leq 2\}$ and $S_1 = S \cup \{(2,2)\}$. Consider the optimization problem*

$$\begin{aligned} &\textit{Min} && (x_1 - 4)^2 + (x_2 - 4)^2 \\ &\textit{subject to} \\ & && (x_1, x_2) \in Conv(S_1). \end{aligned}$$

1. *Express the above as a standard QPP.*
2. *Solve the QPP so obtained graphically.*
3. *Write the KKT system and justify your answer, as obtained at (2) above, by using KKT theorem.*

7.22 *Given two points* $P : (x_1, y_1)$ *and* $Q : (x_2, y_2)$ *in the co-ordinate plane* $\mathbf{R}^2$, *there are three standard metrices which are used for defining the distance between points P and Q. These are*

$$d_1(P,Q) = |x_1 - x_2| + |y_1 - y_2|$$
$$d_2(P,Q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$
$$d_\infty(P,Q) = max(|x_1 - x_2|, |y_1 - y_2|)$$

Let there be four facilities located at points (1,2), (-2,4), (2,6) and (-6,3). A new facility is to be located at a point $P : (x, y)$ *so that the sum of its distance from the four existing facilities is minimum. Formulate the above optimization problem when the distance is taken in the sense* d_1, d_2 *and* d_∞ *respectively.*
Extend the definitions of d_1, d_2 *and* d_∞ *for points P and Q in* $\mathbf{R}^n$.

7.23 *Solve the following QPP by Wolfe's method*

$$\text{Min} \quad (x_1 - 3)^2 + (x_2 - 3)^2$$

subject to

$$x_1 + x_2 \leq 2$$
$$x_1 - x_2 \leq 1$$
$$x_1, x_2 \geq 0.$$

7.24 *In a resource allocation problem, let* $b_i (i = 1, 2, \ldots, m)$ *denote the maximum availability of the* i^{th} *resource and* a_{ij} *denote the units of* i^{th} *resource used in producing one unit of the* j^{th} *product,*$(i = 1, 2, \ldots, m\ ; j = 1, 2, \ldots, n)$. *Let per unit cost of production for the* j^{th} *product be proportional to the units of the* j^{th} *product produced with the constant of proportionality as* c_j $(j = 1, 2, \ldots, n)$.

Formulate the above as an optimization problem and identify if its is a LPP or a QPP.

7.25 *Are the following statements true? Give reasons for your answer.*

- *The quadratic form* $4x_1^2 + 6x_2^2 - 8x_1x_2$ *is positive definite.*
- *The quadratic form* $x_1^2 + x_2^2 + 2x_1x_2$ *is positive definite.*
- *For a skew symmetric matrix, the value of the quadratic form* x^TQx *is always zero.*
- *Let Q be a* $(n \times n)$ *indefinite matrix. Then there certainly exists a point* $\bar{x} \in \mathbf{R}^n$ *such that* $\bar{x}^TQ\bar{x} = 0$.
- *For a negative-definite matrix Q, the matrix* Q^2 *is also negative definite.*

8
Optimality Conditions and Duality in Nonlinear Programming

8.1 Introduction

Taking motivation from the classical calculus methods to optimize a function of one or more real variables, in Chapter 7, we have already given an elementary introduction of the *Karush-Kuhn-Tucker*(KKT) optimality conditions for a general nonlinear programming problem. In this chapter we essentially try to be bit more formal mathematically and give a formal proof of the KKT necessary/sufficient optimality conditions, and use the same to construct the standard Wolfe dual of the given problem. In this context we shall like to remark that this aspect of nonlinear programming, if done properly, is mathematically very involved as it uses many advanced tools of convex analysis and generalized gradients. In our presentation here, we do not plan to go into these details and keep ourselves at a some what lower level in terms of using mathematical tools by assuming that the objective and constraint functions of the given nonlinear programming problem are continuously differentiable.

8.2 Feasible Directions and Linearizing Cone

Consider the nonlinear programming problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \leq 0, \ \ (i = 1, 2, \dots, m)\,. \end{aligned} \tag{8.1}$$

Here $x \in \mathbf{R}^n$ and it is assumed that the feasible region S of problem (8.1) is contained in some open subset D of $\mathbf{R}^n$ where f and g_i $(i = 1, 2, \dots, m)$, are defined and continuously differentiable.

Definition 8.2.1 (Local Min Point/ Global Min Point). *A point $\bar{x} \in S$ is called a local min point of the nonlinear programming problem (8.1) if $\exists\ \delta > 0$ such that $f(\bar{x}) \leq f(x)$, $\forall x \in N_\delta(\bar{x}) \cap S$.*

If the above inequality holds for all $x \in S$, then $\bar{x}$ is called a global min point of the given nonlinear programming problem.

Definition 8.2.2 (Feasible Direction). *Let x be a feasible point, i.e. $x \in S$. We define a feasible direction at x to be any direction d (i.e a vector $d \in \mathbf{R}^n$) with the property that $x + \alpha\, d$ is in the feasible set S for some α sufficiently small, i.e. a direction $d \in \mathbf{R}^n$ is feasible at $x \in S$ if $\exists\ \sigma > 0$ such that $x + \alpha\, d \in S$, $\forall\ 0 < \alpha \leq \sigma$.*

We shall denote by $D(x)$ the set of all feasible directions at x, i.e.

$$D(x) = \left\{ d \in \mathbf{R}^n : \exists\ \sigma > 0 \ \ni\ 0 < \alpha \leq \sigma \Rightarrow x + \alpha\, d \in S \right\}.$$

Thus a direction d is in $D(x)$ if we can move a small distance from x in the direction of d and still remain in the feasible set. Obviously if $x \in Int\ S$ then $D(x) = \mathbf{R}^n$. In Fig 8.1

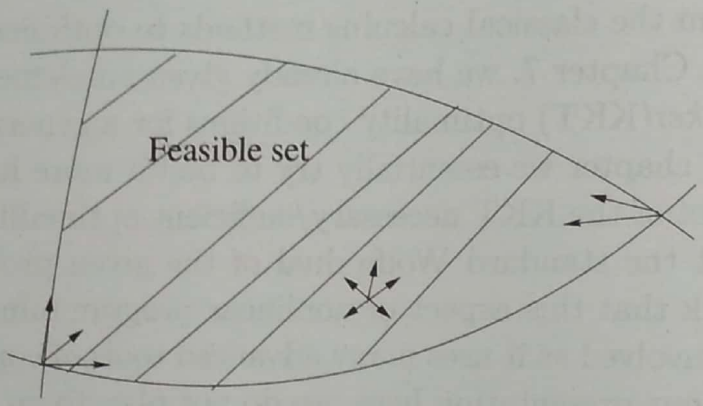


Fig. 8.1.

arrows indicate the feasible directions at the given point.

Our aim here is to establish certain necessary/sufficient optimality conditions for a point $\bar{x}$ to be a local min/ global min point of the given nonlinear programming problem (8.1). In this development, the set $D(x)$ as introduced above will play a major role. We shall be presenting results for the minimization case, the maximization case will follow analogously.

Lemma 8.2.1 *Let $\bar{x}$ be a local min point of problem (8.1). Then $d^T\ \nabla f(\bar{x}) \geq 0$ for all $d \in D(\bar{x})$.*

Proof. If possible, let $\exists\ d_1 \in D(\bar{x})$ with $d_1^T\ \nabla f(\bar{x}) < 0$. Noting that $d_1^T\ \nabla f(\bar{x})$ is the directional derivative of the function f at $\bar{x}$ in the direction of d_1, this implies that

$$\lim_{\theta \to 0} \frac{f(\bar{x} + \theta\, d_1) - f(\bar{x})}{\theta} < 0\,. \tag{8.2}$$

Now using the definition of limit, (8.2) means that $\exists\ \delta > 0$ such that

$$\frac{f(\bar{x} + \theta\, d_1) - f(\bar{x})}{\theta} < 0$$

whenever $-\delta < \theta < \delta,\ \theta \neq 0$.

Therefore, in particular, $\exists\ \delta > 0$ such that

$$f(\bar{x} + \theta\, d_1) - f(\bar{x}) < 0,\ \forall\ 0 < \theta < \delta,$$

i.e.

$$f(\bar{x} + \theta\, d_1) < f(\bar{x}),\ \forall\ 0 < \theta < \delta.$$

But this contradicts that $\bar{x}$ is a local min point of (8.1). □

Corollary 8.2.1 *Let $\bar{x}$ be a local min point of (8.1). Then $d^T \nabla f(\bar{x}) \geq 0$ for all $d \in \overline{D}(\bar{x})$. Here $\overline{D}(\bar{x})$ denotes the closure of the set $D(\bar{x})$.*

Proof. Any direction $d \in \overline{D}(\bar{x})$ may be expressed as the limit of directions $d^{(k)}$ of $D(\bar{x})$, i.e.

$$d \in \overline{D}(\bar{x}) \Rightarrow \exists\ \{d^{(k)}\},\ d^{(k)} \in D(\bar{x}) \text{ such that } d = \lim_{k \to \infty} d^{(k)}.$$

But by Lemma 8.2.1, $(d^{(k)})^T \nabla f(\bar{x}) \geq 0$ for all $d^{(k)} \in D(\bar{x})$ and hence

$$\lim_{k \to \infty} (d^{(k)})^T \nabla f(\bar{x}) \geq 0$$

i.e.

$$d^T \nabla f(\bar{x}) \geq 0.$$

which proves the corollary. □

Corollary 8.2.2 *In case the local min point $\bar{x}$ of the nonlinear programming problem (8.1) is in the interior of S then $D(\bar{x}) = \mathbf{R}^n$. Hence Lemma 8.2.1 gives $d^T \nabla f(\bar{x}) \geq 0\ \forall d \in \mathbf{R}^n$, which implies that $\nabla f(\bar{x}) = 0$, the well known first order necessary optimality condition.*

Definition 8.2.3 (Usable Direction). *Let $x \in S$. A direction $d \in \mathbf{R}^n$ is said to be a usable direction at x if $\exists\ \sigma > 0$ such that $f(x + \alpha\, d) < f(x)$ for all $0 < \alpha \leq \sigma$.*

Thus a small movement from x along a usable direction d strictly improves the value of the objective function.

Definition 8.2.4 (Usable Feasible Direction). *A direction d which is both feasible and usable at x is said to be a usable feasible direction at x.*

Lemma 8.2.2 *Let f be differentiable at $\hat{x} \in S$ and $\nabla f(\hat{x}) \neq 0$. Then $\hat{d} = -\nabla f(\hat{x})$ is a usable feasible direction at $\hat{x}$ provided $\hat{d}$ is feasible.*

Proof. We have to show that $\hat{d} = -\nabla f(\hat{x})$ is usable at $\hat{x}$. For this, it is enough to show that $(\nabla f(\hat{x}))^T \hat{d} < 0$. But

$$\begin{aligned}(\nabla f(\hat{x}))^T\, \hat{d} &= -\,(\nabla f(\hat{x}))^T\, \nabla f(\hat{x}) \\ &= -\,\|\nabla f(\hat{x})\|^2 < 0\end{aligned}$$

So if $\hat{d}$ is feasible, it is also usable feasible. □

Feasible direction vectors are important in many numerical optimization algorithms as they suggest a direction of movement from the current point $x^{(k)}$.

Definition 8.2.5 (Active Constraints). *Let $\hat{x} \in S$. A constraint is called active at $\hat{x}$ if it holds as an equation at $\hat{x}$.*

Let $I = \{i = 1, 2, \ldots, m\}$ and $I_1(\hat{x}) = \{i \in I : g_i(\hat{x}) = 0\}$. Then $I_1(\hat{x})$ is the collection of all active constraints at $\hat{x}$. Let $I_2(\hat{x}) = I \setminus I_1(\hat{x})$. Therefore for $i \notin I_1(\hat{x})$, $g_i(\hat{x}) < 0$ and hence by using the continuity of g_i (for the specific $i \notin I_1(\hat{x})$), we may move a small distance in any direction from $\hat{x}$ without violating that constraint. This gives $D_{I_2}(\hat{x}) = \mathbf{R}^n$. But $D(\hat{x}) = D_{I_1}(\hat{x}) \cap D_{I_2}(\hat{x}) = D_{I_1}(\hat{x}) \cap \mathbf{R}^n = D_{I_1}(\hat{x})$. Therefore inactive constraints do not contribute to the set $D(\hat{x})$ and hence only active constraints at $\hat{x}$ need to be considered.

Definition 8.2.6 (Linearizing Cone). *Let $\hat{x} \in S$ Then the set $\mathfrak{D}(\hat{x})$ given by $\mathfrak{D}(\hat{x}) = \{d \in \mathbf{R}^n : d^T \nabla g_i(\hat{x}) \le 0,\ i \in I_1(\hat{x})\}$ is called the linearizing cone of S at $\hat{x}$.*

The set $\mathfrak{D}(\hat{x})$ is appropriately named as the linearizing cone because it is generated by linearizing the active constraints at $\hat{x}$, and $d \in \mathfrak{D}(\hat{x})$, $\alpha \ge 0$, implies that $\alpha d \in \mathfrak{D}(\hat{x})$.

Theorem 8.2.1 *Let $\hat{x} \in S$. Then $D(\hat{x}) \subset \mathfrak{D}(\hat{x})$.*

Proof. Let $d \in D(\hat{x})$ and $g_i(\hat{x}) = 0$, i.e. $i \in I_1(\hat{x})$. If possible let $d^T \nabla g_i(\hat{x}) > 0$ for some $i \in I_1(\hat{x})$. Then this will mean that $\exists\ \delta > 0$ such that $g_i(\hat{x} + \theta d) > g_i(\hat{x})$, $0 < \theta \le \delta$, But $g_i(\hat{x}) = 0$ and hence

$$g_i(\hat{x} + \theta d) > 0, \quad 0 < \theta \le \delta\,.$$

which implies that $(\hat{x} + \theta d) \notin S$.
Hence $d \notin D(\hat{x})$. Therefore for all $d \in D(\hat{x})$,

$$d^T \nabla g_i(\hat{x}) \le 0\,, \quad \forall i \in I_1(\hat{x}).$$

Thus

$$d \in D(\hat{x}) \Rightarrow d^T \nabla g_i(\hat{x}) \le 0\,, \quad i \in I_1(\hat{x}).$$

Hence $D(\hat{x}) \subset \mathfrak{D}(\hat{x})$. □

Corollary 8.2.3 *Let $\hat{x} \in S$. Then $\overline{D}(\hat{x}) \subset \mathfrak{D}(\hat{x})$.*

Proof We note that $\mathfrak{D}(\hat{x})$ is a closed set and hence $D(\hat{x}) \subset \mathfrak{D}(\hat{x})$ means that $\overline{D}(\hat{x}) \subset \mathfrak{D}(\hat{x})$. □

We now present some examples to illustrate that $\overline{D}(\hat{x})$ may not be equal to $\mathfrak{D}(\hat{x})$.

Example 8.2.1 *Let the constraints be*

$$g_1(x) = x_1^3 - x_2 \leq 0$$
$$g_2(x) = x_1^3 + x_2 \leq 0,$$

and $\hat{x} = (0,0)^T \in S$ *be the given feasible point. Show that* $\overline{D}(\hat{x}) \neq \mathfrak{D}(\hat{x})$.

Solution First we note that both constraints are active at $(0,0)^T$ and

$$\nabla g_1(x)\Big|_{(0,0)} = \begin{pmatrix} 3x_1^2 \\ -1 \end{pmatrix}\Big|_{(0,0)} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$
$$\nabla g_2(x)\Big|_{(0,0)} = \begin{pmatrix} 3x_1^2 \\ 1 \end{pmatrix}\Big|_{(0,0)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Therefore

$$\begin{aligned}\mathfrak{D}(\hat{x}) &= \left\{d \in \mathbf{R}^2 : d^T \nabla g_i(x^{(0)}) \leq 0,\ i = 1,2\right\} \\ &= \left\{(d_1, d_2) : -d_2 \leq 0, d_2 \leq 0\right\} \\ &= \left\{(d_1, d_2) : d_2 = 0\right\} \\ &= \left\{(d_1, 0) : d_1 \in \mathbf{R}\right\}.\end{aligned}$$

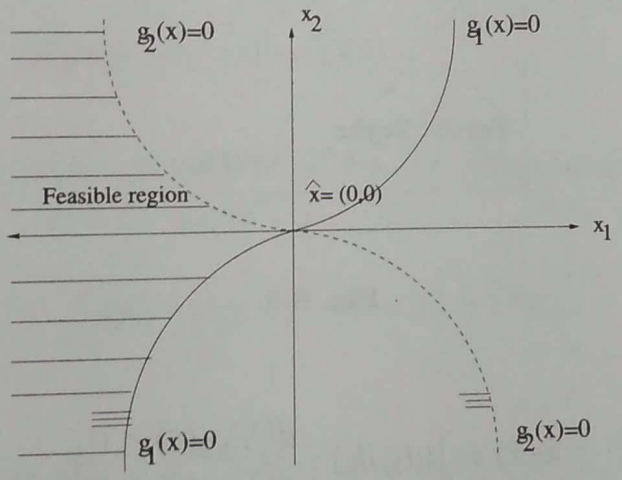


Fig. 8.2.

Now we proceed to obtain $D(\hat{x})$. As $\hat{x} + \theta d = (\theta d_1, \theta d_2)^T$, we have by the definition of $D(\hat{x})$,

$$D(\hat{x}) = \left\{(d_1, d_2) : \theta^3 d_1^3 - \theta d_2 \leq 0, \theta^3 d_1^3 + \theta d_2 \leq 0 \text{ and } 0 < \theta \leq \delta\right\}$$

$$= \left\{(d_1, d_2) : \theta^2 d_1^3 - d_2 \leq 0, \theta^2 d_1^3 + d_2 \leq 0 \text{ and } 0 < \theta \leq \delta\right\}.$$

But $\theta^2 d_1^3 - d_2 \leq 0$ and $\theta^2 d_1^3 + d_2 \leq 0$ imply that $2\theta^2 d_1^3 \leq 0$ which gives $d_1 \leq 0$ as $\theta > 0$. Also, $d_2 \leq -\theta^2 d_1^3$ and $d_2 \geq \theta^2 d_1^3$, i.e $\theta^2 d_1^3 \leq d_2 \leq -\theta^2 d_1^3$. Therefore taking the limit $\theta \to 0^+$, we get $d_2 = 0$. Therefore,

$$D(\hat{x}) = \{(d_1, d_2) : d_1 \leq 0,\ d_2 = 0\},$$

which we can also visualize from Fig. 8.2. Therefore $D(\hat{x}) \subset \mathfrak{D}(\hat{x})$. But $(1,0)^T \in \mathfrak{D}(\hat{x})$ which is not in $D(\hat{x})$ and hence $D(\hat{x}) \neq \mathfrak{D}(\hat{x})$. So a direction d of $\mathfrak{D}(\hat{x})$ may point in an infeasible direction. Here we may note that $\overline{D}(\hat{x}) \neq \mathfrak{D}(\hat{x})$, as $\overline{D}(\hat{x}) = D(\hat{x})$ and $D(\hat{x}) \neq \mathfrak{D}(\hat{x})$.

Example 8.2.2 *Let the constraints be*

$$\begin{aligned} g_1(x) &= -x_1 \leq 0 \\ g_2(x) &= -x_2 \leq 0 \\ g_3(x) &= -(1 - x_1)^3 + x_2 \leq 0, \end{aligned}$$

and $\hat{x} = (1,0)^T \in S$ *be the given feasible point. Show that* $\overline{D}(\hat{x}) \neq \mathfrak{D}(\hat{x})$.

Solution As $g_1(\hat{x}) \neq 0, g_2(\hat{x}) = 0 = g_3(\hat{x})$, constraints g_2 and g_3 are active at $\hat{x}$. Now,

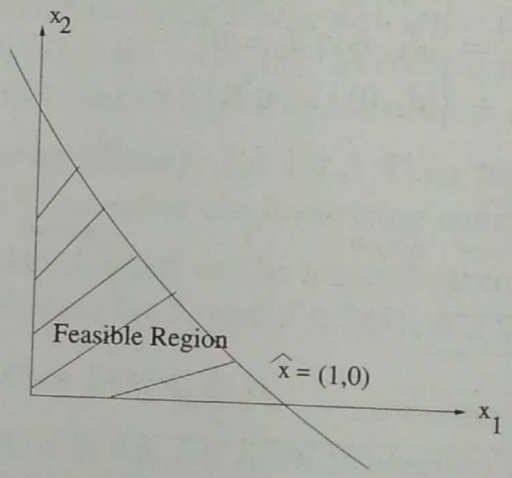


Fig. 8.3.

$$\begin{aligned} \mathfrak{D}(\hat{x}) &= \left\{(d_1, d_2) : \begin{matrix} d^T \nabla g_2(\hat{x}) \leq 0 \\ d^T \nabla g_3(\hat{x}) \leq 0 \end{matrix}\right\} \\ &= \{(d_1, d_2) : -d_2 \leq 0, d_2 \geq 0\} \\ &= \{(d_1, d_2) : d_2 = 0\}. \end{aligned}$$

Also $\hat{x} + \theta d = (1 + \theta d_1, \theta d_2)^T$ and therefore

$$D(\hat{x}) = \{(d_1, d_2) : 1 + \theta d_1 \geq 0, \theta d_2 \geq 0, \theta^3 d_1^3 + \theta d_2 \leq 0,\ 0 < \theta \leq \delta\}$$

But $\theta d_2 \geq 0$ implies that $d_2 \geq 0$ and then $\theta^2 d_1^3 \leq -d_2 \leq 0$ gives $d_1 \leq 0$. Further $d_2 \leq -\theta^2 d_1^3$, i.e. $0 \leq d_2 \leq -\theta^2 d_1^3$ which on taking limit as $\theta \to 0^+$ gives $d_2 = 0$. Therefore

$$D(\hat{x}) = \{(d_1, d_2) : d_1 \leq 0, d_2 = 0\},$$

which is also equal to $\overline{D}(\hat{x})$. Obviously, $\overline{D}(\hat{x}) \subset \mathfrak{D}(\hat{x})$ but the converse inclusion is not true, e.g the point $(2,0)^T \in \mathfrak{D}(\hat{x})$ but it does not belong to $\overline{D}(\hat{x})$.

Remark 8.2.1 *The definition of the linearizing cone $\mathfrak{D}(\hat{x})$ depends explicitly on the constraint functions $g_i (i \in I)$. But the same feasible set may have more than one representation and hence it is possible that $\overline{D}(\hat{x}) \neq \mathfrak{D}(\hat{x})$ for one representation of the given feasible region but $\overline{D}(\hat{x}) = \mathfrak{D}(\hat{x})$ for another representation of the same feasible region. Examples 8.2.3 and 8.2.4 illustrate this point.*

Example 8.2.3 *The two representations (i) $x_1 = 0, x_2 \in \mathbf{R}$ and (ii) $x_1^2 = 0, x_2 \in \mathbf{R}$ give the same feasible region of $\mathbf{R}^2$. Let $\hat{x} = (0,\ 0)^T$. Identify the representation for which $\overline{D}(\hat{x}) = \mathfrak{D}(\hat{x})$.*

Solution The feasible region is the x_2 axis and $\hat{x}$ is the origin. Now

$$\begin{aligned} D(\hat{x}) &= \left\{(d_1, d_2) : 0 + \theta d_1 \leq 0, -\theta d_1 \leq 0, 0 < \theta \leq \delta\right\} \\ &= \left\{(d_1, d_2) : d_1 = 0\right\} = \overline{D}(\hat{x}) \end{aligned}$$

Also

$$\begin{aligned} \mathfrak{D}(\hat{x}) &= \left\{(d_1, d_2) : (d_1, d_2)\begin{pmatrix} 1 \\ 0 \end{pmatrix} \leq 0 \text{ and } (d_1, d_2)\begin{pmatrix} -1 \\ 0 \end{pmatrix} \leq 0\right\} \\ &= \left\{(d_1, d_2) : d_1 = 0\right\} = \overline{D}(\hat{x}) \end{aligned}$$

So $D(\hat{x}) = \mathfrak{D}(\hat{x})$.

But if we take the second representation for the same feasible region S, i.e. $x_1^2 = 0$, $x_2 \in \mathbf{R}$ i.e $x_1^2 \leq 0, -x_1^2 \leq 0$, $x_2 \in \mathbf{R}$, then

$$D(\hat{x}) = \left\{(d_1, d_2) : d_1 = 0\right\} = \overline{D}(\hat{x})$$

and

$$\begin{aligned} \mathfrak{D}(\hat{x}) &= \left\{(d_1, d_2) : (d_1, d_2)\begin{pmatrix} 2x_1 \\ 0 \end{pmatrix}\Big|_{(0,0)} \leq 0 \text{ and } (d_1, d_2)\begin{pmatrix} -2x_1 \\ 0 \end{pmatrix}\Big|_{(0,0)} \leq 0\right\} \\ &= \left\{(d_1, d_2) : d_1.0 + d_2.0 = 0\right\} = \mathbf{R}^2. \end{aligned}$$

So $\overline{D}(0,0) \neq \mathfrak{D}(0,0)$, although the feasible region S remains the same, namely the x_2-axis.

Example 8.2.4 *Let (i) $x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0$ and (ii) $(1 - x_1 - x_2)^3 \geq 0, x_1 \geq 0$, $x_2 \geq 0$ be the two different sets of constraints. Verify that both sets of constraints give the same feasible region. Let $\hat{x} = (1/2, 1/2)^T$. Identify the representation for which $\overline{D}(\hat{x}) = \mathfrak{D}(\hat{x})$.*

Solution It is simple to verify that the feasible region given by both sets of constraints is the same solid triangle with vertices as $(0,0), (1,0)$ and $(0,1)$. Now for the representation given by the first set of constraints

$$\mathfrak{D}(1/2,1/2) = \left\{(d_1,d_2) : d_1 + d_2 \leq 0\right\},$$

and

$$\begin{aligned} D(1/2,1/2) &= \left\{(d_1,d_2) : \hat{x} + \theta d \in S, 0 < \theta \leq \delta\right\} \\ &= \left\{(d_1,d_2) : (1/2 + \theta d_1) + (1/2 + \theta d_2) \leq 1, -(1/2 + \theta d_1) \leq 0,\right. \\ &\quad \left. -(1/2 + \theta d_2) \leq 0, 0 < \theta \leq \delta\right\} \\ &= \left\{(d_1,d_2) : \theta(d_1 + d_2) \leq 0, 0 < \theta \leq \delta\right\} \\ &= \left\{(d_1,d_2) : d_1 + d_2 \leq 0\right\} = \overline{D}(1/2,1/2)\,. \end{aligned}$$

Therefore, $\overline{D}(1/2,1/2) = \mathfrak{D}(1/2,1/2)$.
However for the second representation, i.e. for the second set of constraints $\overline{D}(1/2,1/2) \neq \mathfrak{D}(1/2,1/2)$.

8.3 Basic Constraint Qualification

We now define the basic constraint qualification for nonlinear programming problem (8.1).

Definition 8.3.1 (Basic Constraint Qualification). *Given the non-linear programming problem (8.1), we say that the constraints satisfy the basic constraint qualification at a feasible point $\hat{x}$ if $\overline{D}(\hat{x}) = \mathfrak{D}(\hat{x})$.*

Lemma 8.3.1 *Let (i) $\overline{x}$ be a local min point of the given nonlinear programming problem and (ii) the basic constraint qualification holds at $\overline{x}$, i.e $\overline{D}(\overline{x}) = \mathfrak{D}(\overline{x})$. Then*

$$d^T \nabla f(\overline{x}) \geq 0, \quad \forall d \in \mathfrak{D}(\overline{x}).$$

The proof follows from Corollary 8.2.1 simply by replacing $\overline{D}(\overline{x})$ by $\mathfrak{D}(\overline{x})$.

We now introduce two more sets $Z_1(\hat{x})$ and $Z_2(\hat{x})$, for any feasible point $\hat{x}$, as follows

$$Z_1(\hat{x}) = \left\{d \in \mathbf{R}^n : d^T \nabla f(\hat{x}) < 0\right\},$$

and

$$\begin{aligned} Z_2(\hat{x}) &= Z_1(\hat{x}) \cap \mathfrak{D}(\hat{x}) \\ &= \left\{d \in \mathbf{R}^n : d^T \nabla g_i(\hat{x}) \leq 0, i \in I_1(\hat{x}), d^T \nabla f(\hat{x}) < 0\right\}. \end{aligned}$$

Then Lemma 8.2.1 can be restated as Lemma 8.3.2 given below.

Lemma 8.3.2 *Let $\bar{x}$ be a local min point of the given nonlinear programming problem (8.1) and $\overline{D}(\bar{x}) = \mathfrak{D}(\bar{x})$. Then $Z_2(\bar{x}) = \phi$.*

Remark 8.3.1 *Here it may be remarked that if for a feasible point $\bar{x}$, $\overline{D}(\bar{x}) = \mathfrak{D}(\bar{x})$ and $Z_2(\bar{x}) = \phi$, then it does not necessarily imply that $\bar{x}$ is a local min point of problem (8.1). Example 8.3.1 given below illustrates this point.*

Example 8.3.1 *Consider the problem*

Min $-x_2$

subject to

$$x_1^2 + x_2^2 \leq 4$$
$$-x_1^2 + x_2 \leq 0\,.$$

Let $\bar{x} = (0, \quad 0)^T$. Show that $Z_2(\bar{x}) = \phi$, $\overline{D}(\bar{x}) = \mathfrak{D}(\bar{x})$, but $\bar{x}$ is a not a local min point of the given problem.

Solution For the point $\bar{x} = (0,0)^T$, we have

$$\nabla f(\bar{x}) = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad \nabla g_1(\bar{x}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \nabla g_2(\bar{x}) = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and g_2 is active at $\bar{x} = (0,0)^T$. Therefore

$$D(0,0) = \left\{(d_1, d_2) : -\theta^2 d_1^2 + \theta d_2 \leq 0, \theta^2(d_1^2 + d_2^2) \leq 4, 0 < \theta \leq \delta\right\}$$

i.e.

$$D(\bar{x}) = \left\{(d_1, d_2) : d_2 \leq \theta d_1^2, 0 < \theta \leq \delta\right\} = \overline{D}(0,0)\,.$$

Also,

$$\mathfrak{D}(0,0) = \left\{(d_1, d_2) : d^T \nabla g_2(\bar{x}) \leq 0\right\}$$

$$= \left\{(d_1, d_2) : d_2 \leq 0\right\}.$$

Therefore $\mathfrak{D}(0,0) \subset D(0,0) = \overline{D}(0,0)$. As for feasible x, $\overline{D}(x) \subset \mathfrak{D}(x)$ is always true, we have $\mathfrak{D}(0,0) = \overline{D}(0,0)$.

Further

$$\begin{aligned} Z_2(0,0) = Z_2(\bar{x}) &= \left\{(d_1, d_2) : d^T \nabla g_2(\bar{x}) \leq 0, d^T \nabla f(\bar{x}) < 0\right\} \\ &= \left\{(d_1, d_2) : d_2 \leq 0, -d_2 < 0\right\} \\ &= \left\{(d_1, d_2) : d_2 \leq 0, d_2 > 0\right\} \\ &= \phi. \end{aligned}$$

But $\bar{x} = (0,0)^T$ is not a local min point of the objective function $f(x) = -x_2$, as can be seen from Fig. 8.4.

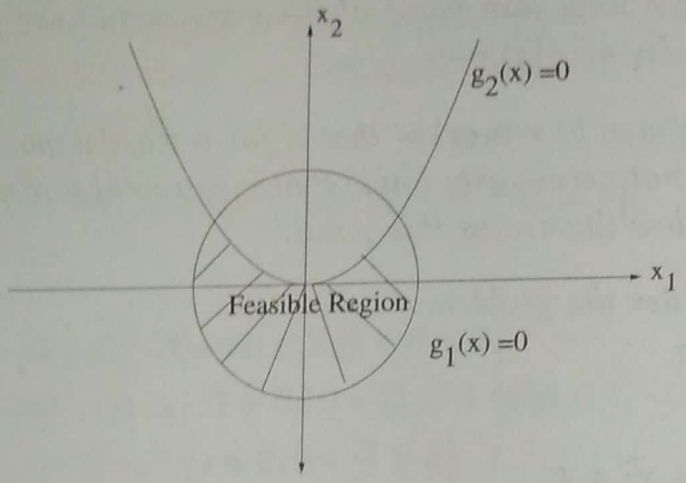


Fig. 8.4.

8.4 Lagrangian and Lagrange Multipliers

Definition 8.4.1 (Lagrangian or Lagrange Function). *Given the nonlinear programming problem (8.1), the function* $L : \mathbf{R}^n \times \mathbf{R}^m_+ \to \mathbf{R}$ *given by* $L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$ *is called its Lagrangian or Lagrange function. Further the vector* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ *is called the vector of Lagrange multipliers.*

The following theorem gives the necessary and sufficient conditions for the existence of Lagrange multipliers at a feasible point $\hat{x}$.

Theorem 8.4.1 (Existence of Lagrange Multipliers).
Let S *be the feasible region of the given nonlinear programming problem (8.1). Let* $\hat{x} \in S$. *Then* $Z_2(\hat{x}) = \phi$ *if and only if there exists a vector of Lagrange multipliers* $\hat{\lambda} \in \mathbf{R}^m_+$, *such that*
(i) $\nabla f(\hat{x}) + \sum_{i \in I} \hat{\lambda}_i \nabla g_i(\hat{x}) = 0$
(ii) $g_i(\hat{x}) \leq 0, \ i \in I$
(iii) $\hat{\lambda}_i g_i(\hat{x}) = 0, \ i \in I$
(iv) $\hat{\lambda}_i \geq 0, \ i \in I.$

Remark 8.4.1 *The above system (i)-(iv) is called the KKT system corresponding to a given feasible point* $\hat{x}$. *If, for a given point* $\hat{x} \in S$, *there exists* $\hat{\lambda}_i$ $(i \in I)$ *such that* $(\hat{x}, \hat{\lambda}_1, \ldots, \hat{\lambda}_m)$ *satisfies the KKT system then we say that at the point* $\hat{x}$, *the KKT conditions hold or* $\hat{x}$ *is a KKT point. Thus Theorem 8.4.1 gives the necessary and sufficient condition for* $\hat{x}$ *to be a KKT point.*

In proving Theorem 8.4.1, we shall make use of the following lemma called the Farkas' Lemma.

Lemma 8.4.1 (Farkas' Lemma). *Let $a^{(0)}, a^{(1)}, \ldots, a^{(r)}$ be $(r+1)$ vectors in $\mathbf{R}^n$. Then $\exists$ scalars $\beta_k \geq 0 \quad (k = 1, 2, \ldots, r)$ such that*

$$a^{(0)} = \sum_{k=1}^{r} \beta_k a^{(k)},$$

if and only if $y^T a^{(0)} \geq 0$ for all y satisfying $y^T a^{(k)} \geq 0$, $(k = 1, 2, \ldots, r)$.

Thus if $A = [a^{(1)}, \ldots, a^{(r)}]_{n \times r}$ and $T = \left\{ y \in \mathbf{R}^n : y^T A \geq 0 \right\}$, then the Farkas' Lemma states that

$$(y \in T \Rightarrow y^T a^{(0)} \geq 0) \Leftrightarrow \exists \quad \beta \in \mathbf{R}^r_+ \text{ such that } a^{(0)} = A^T \beta \,.$$

Proof. Let $a^{(0)} = A^T \beta = \sum_{k=1}^{r} \beta_k a^{(k)}$, $\beta \in \mathbf{R}^r_+$. Then

$$y^T a^{(0)} = \sum_{k=1}^{r} \beta_k y^T a^{(k)} \geq 0.$$

Conversely let $y^T a^{(0)} \geq 0$ for all $y \in T$. We wish to show that $a^{(0)}$ is a non-negative linear combination of vectors $a^{(k)}$, $(k = 1, 2, \ldots, r)$. For this, consider the LPP

$$\begin{aligned} &\text{Min} && y^T a^{(0)} \\ &\text{subject to} \\ & && y^T a^{(k)} \geq 0, \ (k = 1, 2, \ldots, r) \,, \end{aligned} \tag{8.3}$$

and note that the linear programming problem (8.3) has an optimal solution $y = 0$ as for all $y \in T$, $y^T a^{(0)} \geq 0$. Therefore from the duality theorem of linear programming, the dual of the above problem (8.3) is feasible and has a finite optimal solution. But the dual of (8.3) is

$$\begin{aligned} &\text{Max} && 0^T \beta \\ &\text{subject to} \\ & && a^{(0)} = \sum_{k=1}^{r} \beta_k a^{(k)} \\ & && \beta_k \geq 0, \quad (k = 1, 2, \ldots, r) \,. \end{aligned} \tag{8.4}$$

Thus there exists $\beta \in \mathbf{R}^r_+$ such that $a^{(0)} = A^T \beta$. □

Now we give the proof of Theorem 8.4.1. As mentioned earlier, we shall make use of the Farkas' lemma for giving this proof.

Proof of the Theorem 8.4.1.
We first note that $\mathfrak{D}(\hat{x}) \neq \phi$ because $0 \in \mathfrak{D}(\hat{x})$. Therefore $Z_2(\hat{x}) = \phi$ if and only if $\forall d \in \mathfrak{D}(\hat{x})$ we have

$$d^T \nabla f(\hat{x}) \geq 0,$$

i.e.

$$d^T \nabla g_i(\hat{x}) \leq 0,\ i \in I_1(\hat{x}) \Rightarrow d^T \nabla f(\hat{x}) \geq 0.$$

If we now take $\hat{a} = \nabla f(\hat{x})$ and $a^{(i)} = -\nabla g_i(\hat{x}),\ i \in I(\hat{x})$, then the Farkas' Lemma becomes applicable and therefore there exist multipliers $\hat{\lambda}_i \geq 0,\ (i \in I_1(\hat{x}))$ such that

$$\nabla f(\hat{x}) = \sum_{i \in I_1(\hat{x})} \hat{\lambda}_i (-\nabla g_i(\hat{x}))$$

i.e.

$$\nabla f(\hat{x}) + \sum_{i \in I_1(\hat{x})} \hat{\lambda}_i \nabla g_i(\hat{x}) = 0. \tag{8.5}$$

Now taking $\hat{\lambda}_i = 0$ for $i \notin I_1(\hat{x})$ we have from (8.5)

$$\nabla f(\hat{x}) + \sum_{i \in I} \hat{\lambda}_i \nabla g_i(\hat{x}) = 0\ .$$

Also

$$g_i(\hat{x}) \leq 0,\ (i \in I), \quad \hat{\lambda}_i \geq 0,\ (i \in I)\ .$$

Further the relation $\hat{\lambda}_i \nabla g_i(\hat{x}) = 0,\ (i \in I)$ follows because for $i \in I_1(\hat{x}), g_i(\hat{x}) = 0$ and for $i \notin I_1(\hat{x})$, as per choice, $\hat{\lambda}_i = 0$. This proves the main theorem. □

8.5 Karush-Kuhn-Tucker Necessary/Sufficient Optimality Conditions

We now state the most basic result in nonlinear programming, namely the Karush-Kuhn-Tucker (KKT) Theorem. As most of the mathematical concepts needed to prove this theorem have already been presented in earlier sections, the proof will follow simply by recalling some of the earlier results.

Theorem 8.5.1 (Necessary Part of the KKT Theorem).
Let $\overline{x}$ be a local min point of the given nonlinear programming problem (8.1) at which the basic constraint qualification holds, i.e $\overline{D}(\overline{x}) = \mathfrak{D}(\overline{x})$. Then there exist multipliers (called KKT multipliers) $\overline{\lambda}_i$ $(i \in I)$ such that the following KKT conditions hold
(i) $\nabla f(\overline{x}) + \sum_{i \in I} \overline{\lambda}_i \nabla g_i(\overline{x}) = 0$
(ii) $g_i(\overline{x}) \leq 0$ $(i \in I)$
(iii) $\overline{\lambda}_i g_i(\overline{x}) = 0$ $(i \in I)$
(iv) $\overline{\lambda}_i \geq 0,\ (i \in I)$.

Proof. From the given hypothesis we infer from Lemma 8.3.2 that $Z_2(\bar{x}) = \phi$. The proof then follows directly by employing Theorem 8.4.1. □

Theorem 8.5.2 (Sufficient Part of the KKT Theorem). *Let $(\bar{x}, \bar{\lambda}_1, \ldots, \bar{\lambda}_m)$ satisfy the KKT conditions (i)-(iv) of the Theorem 8.5.1 above. Let f, and g_i $(i \in I)$ be differentiable convex functions. Then $\bar{x}$ is a global min point of the given nonlinear programming problem (8.1).*

Proof. We have to prove that $\bar{x}$ is a global min point of problem (8.1), i.e. $f(\bar{x}) \leq f(x)$ for all feasible x. For this let x be any feasible point. Then by the convexity of f we have

$$f(x) - f(\bar{x}) \geq (x - \bar{x})^T \nabla f(\bar{x}) . \tag{8.6}$$

Now substituting for $\nabla f(\bar{x})$ from the KKT condition (i), the above inequality gives

$$f(x) - f(\bar{x}) \geq -\sum_{i=1}^{m} \bar{\lambda}_i ((x - \bar{x})^T \nabla g_i(\bar{x})) . \tag{8.7}$$

But g_i $(i \in I)$ is a convex function and hence

$$g_i(x) - g_i(\bar{x}) \geq (x - \bar{x})^T \nabla g_i(\bar{x}),$$

i.e.

$$-(x - \bar{x})^T \nabla g_i(\bar{x}) \geq g_i(\bar{x}) - g_i(x). \tag{8.8}$$

As $\bar{\lambda}_i \geq 0$ $(i \in I)$, (8.7) and (8.8) gives

$$f(x) - f(\bar{x}) \geq -\sum_{i=1}^{m} \bar{\lambda}_i g_i(\bar{x}) - \sum_{i=1}^{m} \bar{\lambda}_i g_i(x). \tag{8.9}$$

Also from the KKT condition (iii) we have $\bar{\lambda}_i g_i(\bar{x}) = 0$, $(i \in I)$. Further the feasibility of x gives $g_i(x) \leq 0$ $(i \in I)$. Therefore from (8.9) we get

$$f(x) - f(\bar{x}) \geq -\sum_{i=1}^{m} \bar{\lambda}_i g_i(x) \geq 0,$$

i.e.

$$f(x) - f(\bar{x}) \geq 0,$$

i.e. $f(\bar{x}) \leq f(x)$ for all feasible x. □

Remark 8.5.1 *The KKT conditions, in general, are not sufficient for a point $\bar{x}$ to be a local min point/global min point and some convexity type assumptions on f and g_i are certainly needed. Example 8.5.1, given below, is an illustration in this regard.*

Example 8.5.1 *Consider the nonlinear programming problem*

$$\begin{aligned} &\textit{Min} \quad -x_2 \\ &\textit{subject to} \\ &\qquad x_1^2 + x_2^2 \le 4 \\ &\qquad -x_1^2 + x_2 \le 0 \end{aligned}$$

Verify that the KKT conditions are satisfied at (0,0) but it is not a global (not even a local) min point.

Solution At $\hat{x} = (0,0) \in S$ (feasible region), $Z_2(\hat{x}) = \phi$, the KKT conditions are certainly satisfied. This can also be verified by taking $\hat{x} = (0,0)$, $\hat{\lambda}_1 = 0$, $\hat{\lambda}_2 = 1$ in the KKT system for the given problem.
But $(0,0)$ is not a local min point as can be seen from Fig 8.5 because the objective function is to minimize $-x_2$ i.e. to max x_2, so the optimal points are P and Q as shown in Fig 8.5.

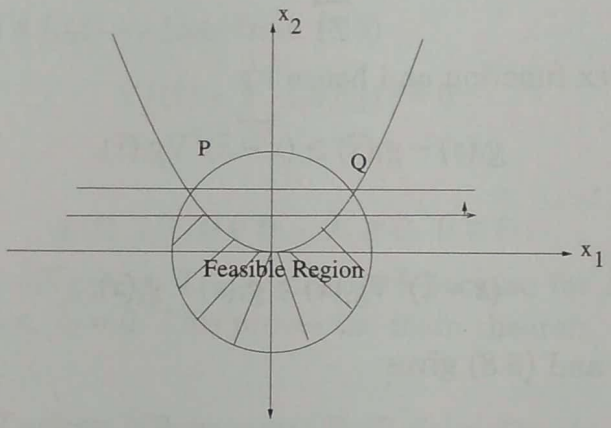


Fig. 8.5.

Remark 8.5.2 *In proving Theorem 8.5.1, the important thing is to guarantee that at a local min point $\bar{x}$, $Z_2(\bar{x}) = \phi$. In our presentation here, we have used the basic constraint qualification $\overline{D}(\bar{x}) = \mathfrak{D}(\bar{x})$ for this purpose. But obviously there could be other 'suitable' constraint qualifications giving $Z_2(\bar{x}) = \phi$. Though the literature contains numerous such constraint qualifications, we discuss only the linear independence constraint qualification below because of the fact that it can possibly be verified easily in comparison to other constraint qualifications. Nevertheless, the topic of constraint qualifications is important and we may refer to Mangasarian [109] for further details in this regard.*

Linear Independence Constraint Qualification

Let $\bar{x}$ be a feasible point of the nonlinear programming problem (8.1) and $I(\bar{x})$ be the index set of active constraints at $\bar{x}$. Then we say that the *linear independence constraint qualification* holds at $\bar{x}$ if the vectors $\nabla g_i(\bar{x})$ $(i \in I(\bar{x}))$, are linearly independent.

Remark 8.5.3 *One certainly needs some constraint qualification so as to guarantee that at a local min point $\bar{x}$, $Z_2(\bar{x}) = \phi$. Otherwise, the KKT conditions may not hold at an optimal point as is illustrated in Example 8.5.2 given below.*

Example 8.5.2 *Consider the nonlinear programming problem*

$$\begin{aligned} &\textit{Min} && -x_1 \\ &\textit{subject to} \\ &&-(1-x_1)^3 + x_2 \le 0 \\ &&-x_1 \le 0 \\ &&-x_2 \le 0. \end{aligned}$$

and verify that (1,0) is optimal. Does basic constraint qualification hold at (1,0)? Are KKT conditions satisfied at (1,0)?

For the above problem $\bar{x} = (1,0)$ is an optimal solution (see Fig 8.6) but the KKT conditions are not holding at $(1,0)$. We may also check that $\overline{D}(\bar{x}) \neq \mathfrak{D}(\bar{x})$. Had this basic constraint qualification or some other 'suitable' constraint qualification been holding at $\bar{x}$, then optimality of $\bar{x}$ would have given $Z_2(\bar{x}) = \phi$ and hence KKT conditions would have certainly been satisfied.

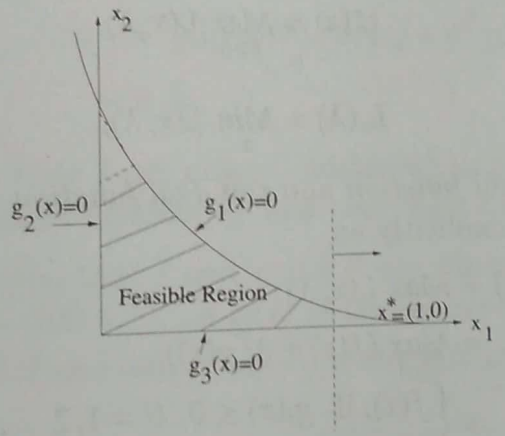


Fig. 8.6.

However, here we have $Z_2(\bar{x}) \neq \phi$. This can be verified because

$$\mathfrak{D}(\bar{x}) = \{(d_1, d_2) : d_2 = 0\} = \{(d_1, 0) : d_1 \in \mathbf{R}\},$$

$$\begin{aligned} Z_1(\bar{x}) &= \left\{(d_1, d_2) : d^T \nabla f(\bar{x}) < 0\right\} \\ &= \left\{(d_1, d_2) : -d_1 < 0\right\} \\ &= \left\{(d_1, d_2) : d_1 > 0\right\}, \end{aligned}$$

and therefore

$$Z_2(\bar{x}) = Z_1(\bar{x}) \cap \mathfrak{D}(\bar{x}) = \{(d_1, 0) : d_1 > 0\} \neq \phi.$$

8.6 Duality in Nonlinear Programming

In this section we present duality in nonlinear programming via the Lagrangian approach. Here we shall restrict our discussion to the convex case only.

Let f, g_i $(i = 1, 2, \ldots, m)$ be real valued convex functions over $\mathbf{R}^n$. We now consider the following convex programming problem (CPP)

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \le 0, \quad i \in I = \{1, 2, \ldots, m\} \end{aligned} \tag{8.10}$$

and associate the Lagrange function L given by

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = f(x) + \lambda^T g(x),$$

where $x \in \mathbf{R}^n$, $\lambda \in \mathbf{R}^m_+$ and $g(x) = (g_1(x), \ldots, g_m(x))^T$.

To motivate the construction of the dual for the convex programming problem (8.10), we introduce the following two functions

$$L^*(x) = \underset{\lambda \ge 0}{Max}\ L(x, \lambda)$$

and

$$L_*(\lambda) = \underset{x}{Min}\ L(x, \lambda),$$

and call them as the *primal function* and the *dual function* respectively. Now the function $L^*(x)$ can be written explicitly as

$$\begin{aligned} L^*(x) &= \underset{\lambda \ge 0}{Max}\ L(x, \lambda) \\ &= \underset{\lambda \ge 0}{Max}\ (f(x) + \lambda^T g(x)) \\ &= \begin{cases} f(x), \text{ if } g_i(x) \le 0 \quad (i = 1, 2, \ldots, m), \\ +\infty, \text{ otherwise .} \end{cases} \end{aligned}$$

Therefore the problem

$$\underset{x}{Min}\ L^*(x),$$

i.e.

$$\underset{x}{Min}\ \underset{\lambda \ge 0}{Max}\ L(x, \lambda), \tag{8.11}$$

comes out to be same as the original problem (8.10).

Looking at the above problem, it becomes natural now to construct the other problem as well, namely,

$$\underset{\lambda \ge 0}{Max}\ L_*(\lambda),$$

i.e.

$$\underset{\lambda \geq 0}{Max}\ \underset{x}{Min}\ L(x, \lambda) \tag{8.12}$$

and see if there is any meaningful connection between these two problems. In the following we shall see that under the assumption of convexity the two problems '$\underset{x}{Min}\ \underset{\lambda \geq 0}{Max}\ L(x, \lambda)$' and '$\underset{\lambda \geq 0}{Max}\ \underset{x}{Min}\ L(x, \lambda)$' are really related exactly the same way as the standard linear programming and its dual are related, and could be taken as the standard primal-dual pair of the convex programming problems.

The above discussion suggests that the dual (8.12) for problem (8.11) or equivalently the given problem (8.10) could be defined as

$$\underset{\lambda \geq 0}{Max}\ L_*(\lambda),$$

i.e.

$$\underset{\lambda \geq 0}{Max}\ \underset{x}{Min}\ L(x, \lambda)$$

i.e.

$$\underset{\lambda \geq 0}{Max}\ \underset{x}{Min}\ (f(x) + \lambda^T g(x)),$$

i.e.

$$\begin{aligned} &\text{Max} && f(u) + \lambda^T g(u) \\ &\text{subject to} \\ & && f(u) + \lambda^T g(u) = \underset{x}{Min}\ (f(x) + \lambda^T g(x)) \\ & && \lambda \geq 0. \end{aligned} \tag{8.13}$$

Remark 8.6.1 *If we wish to be more precise mathematically then in the above discussion, 'max' and 'min' should be replaced by 'sup' and 'inf' respectively.*

We now use the assumption that f and g_i $(i \in I)$ are differentiable convex functions. This implies that $L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$ is also convex in x for all fixed $\lambda \geq 0$. Therefore $\nabla_x L(x, \lambda)\big|_{(\bar{x}, \bar{\lambda})} = 0$ if and only if $L(\bar{x}, \bar{\lambda})$ is the minimum value of $L(x, \lambda)$, i.e.

$$L(\bar{x}, \bar{\lambda}) = \underset{x}{Min}\ [f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)]\ .$$

Hence the dual (8.13) can be rewritten as

$$\begin{aligned} &\text{Max} && f(u) + \sum_{i=1}^{m} \lambda_i g_i(u) \\ &\text{subject to} \\ & && \nabla f(u) + \sum_{i=1}^{m} \lambda_i g_i(u) = 0 \\ & && \lambda_i \geq 0,\ i \in I\ . \end{aligned} \tag{8.14}$$

This dual (8.14) is called the *Wolfe Dual* of the given convex programming problem (8.10). For the sake of convenience, in the following we shall refer to the convex programming problem (8.10) as (CP) and its dual (8.14) as (CD).

Theorem 8.6.1 (Weak Duality Theorem). *Let x be feasible for (CP) and (u, λ) be feasible for (CD). Then*

$$f(x) \geq f(u) + \sum_{i=1}^{m} \lambda_i g_i(x) .$$

Proof. By the convexity of f we have

$$\begin{aligned} f(x) - f(u) &\geq (x-u)^T \nabla_x f(u) \\ &= (x-u)^T \left[-\sum_{i=1}^{m} \lambda_i \nabla g_i(u)\right] \qquad (8.15) \\ &= -\sum_{i=1}^{m} \lambda_i ((x-u)^T \nabla g_i(u)) \qquad (8.16) \\ &\geq \sum_{i=1}^{m} \lambda_i g_i(u) - \sum_{i=1}^{m} \lambda_i g_i(x) \geq \sum_{i=1}^{m} \lambda_i g_i(u) . \qquad (8.17) \end{aligned}$$

Therefore $f(x) \geq f(u) + \sum_{i=1}^{m} \lambda_i g_i(u)$.

Here (8.16) uses the convexity of $g_i (i \in I)$ and the other inequalities follow because of the feasibility of x and (u, λ) for (CP) and (CD) respectively.

Corollary 8.6.1 *Let $\overline{x}$ be feasible for (CP) and $(\overline{u}, \overline{\lambda})$ be feasible for (CD). Further, let $f(\overline{x}) = f(\overline{u}) + \sum_{i=1}^{m} \lambda_i g_i(\overline{u})$. Then $\overline{x}$ is optimal for (CP) and $(\overline{u}, \overline{\lambda})$ is optimal for (CD).*

Proof. Let x be feasible to (CP). Then by Theorem 8.6.1, we have $f(x) \geq f(\overline{u}) + \sum_{i=1}^{m} \lambda_i g_i(\overline{u}) = f(\overline{x})$, which shows that $\overline{x}$ is optimal to (CP). Similarly we can prove that $(\overline{u}, \overline{\lambda})$ is optimal to (CD).

Theorem 8.6.2 (Direct Duality Theorem). *Let $\overline{x}$ be optimal to (CP) and the basic constraint qualification hold at $\overline{x}$. Then there exists $\overline{\lambda}$ such that $(\overline{x}, \overline{\lambda})$ is optimal to (CD). Further the optimal values of the two objective functions coincide.*

Proof. We note that $\overline{x}$ is optimal to (CP) at which basic constraint qualification holds. Therefore by KKT Theorem 8.5.1, there exists $\overline{\lambda}$ such that $(\overline{x}, \overline{\lambda})$ satisfies the KKT conditions

(i) $\nabla f(\overline{x}) + \sum_{i=1}^{m} \overline{\lambda}_i \nabla g_i(\overline{x}) = 0$

(ii) $g_i(\overline{x}) \leq 0$

(iii) $\overline{\lambda}_i g_i(\overline{x}) = 0, \ (i \in I)$

(iv) $\overline{\lambda}_i \geq 0$ $(i \in I)$.

Now equations (i) and (iv) give that $(\overline{x}, \overline{\lambda})$ is feasible to (CD). Also (iii) gives $f(\overline{x}) = f(\overline{x}) + \sum_{i=1}^{m} \overline{\lambda}_i g_i(\overline{x})$ and so $(\overline{x}, \overline{\lambda})$ is optimal to (CD) by Corollary 8.6.1.

We now state several converse duality theorems without proofs.

Theorem 8.6.3 (Strict Converse Duality Theorem). *Let (i) (CP) have an optimal solution $\overline{x}$ (ii) the basic constraint qualification hold at $\overline{x}$ (iii) f be strictly convex and g_i $(i \in I)$ be convex functions (iv) $(\overline{u}, \overline{\lambda})$ be optimal to (CD). Then $\overline{x} = \overline{u}$.*

Theorem 8.6.4 (Hanson's Converse Duality Theorem). *Let f, g_i $(i \in I)$ be twice differentiable convex functions. Let $(\overline{x}, \overline{\lambda})$ be optimal to (CD), at which the basic constraint qualification holds. Let the Hessian $\nabla^2 L(\overline{x}, \overline{\lambda})$ be non-singular. Then $\overline{x}$ is optimal to (CP) and the two optimal values are equal.*

Remark 8.6.2 *In stating various duality theorems we have made use of the basic constraint qualifications defined earlier (Definition 8.3.1). However in view of Remark 8.5.2, rather than taking the basic constraint qualification, we can as well take some other 'suitable' constraint qualification, e.g. Kuhn-Tucker constraint qualification (Mangasarian [109]).*

8.7 Certain Special Cases of Wolfe Dual

(i) *Linear Programming Duality*
Consider the linear programming problem

$$\begin{aligned} &\text{Max} && c^T x \\ &\text{subject to} \\ & && Ax \leq b \\ & && x \geq 0\,. \end{aligned} \tag{8.18}$$

Now we consider this LPP in 'min' form and write its Wolfe dual. So the given LPP is

$$\begin{aligned} &-\text{Min} && -c^T x \\ &\text{subject to} \\ & && Ax - b \leq 0 \\ & && -x \leq 0, \end{aligned}$$

and its Lagrangian is

$$L(x, \lambda) = -c^T x + \lambda^T (Ax - b) - \mu^T x,$$

giving

$$\nabla_x L(x, \lambda) = -c + A^T \lambda - \mu.$$

Hence its Wolfe dual is

$$\begin{aligned} \text{Max} \quad & L(x, \lambda, \mu) \\ \text{subject to} \quad & \\ & \nabla_x L(x, \lambda, \mu) = 0 \\ & \lambda \geq 0 \\ & \mu \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} \text{Max} \quad & -c^T x + \lambda^T (Ax - b) - \mu^T x \\ \text{subject to} \quad & \\ & -c + A^T \lambda - \mu = 0 \\ & \lambda \geq 0 \\ & \mu \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} \text{Max} \quad & -b^T \lambda \\ \text{subject to} \quad & \\ & A^T \lambda = c + \mu \geq c \\ & \lambda \geq 0 . \end{aligned}$$

Therefore the dual of the original problem is

$$\begin{aligned} -\text{Max} \quad & -b^T \lambda \\ \text{subject to} \quad & \\ & A^T \lambda \geq c \\ & \lambda \geq 0. \end{aligned}$$

i.e.

$$\begin{aligned} \text{Min} \quad & b^T \lambda \\ \text{subject to} \quad & \\ & A^T \lambda \geq c \\ & \lambda \geq 0. \end{aligned} \tag{8.19}$$

The linear programming problems (8.18) and (8.19) constitute the usual primal and dual pair of LPP's as obtained in Chapter 4.

(ii) *Quadratic Programming Duality*

Let the problem be

$$\begin{aligned} \text{Max} \quad & c^T x - \frac{1}{2} x^T D x \\ \text{subject to} \quad & \\ & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{8.20}$$

i.e.

$$\begin{aligned} -\text{Min} \quad & -c^T x + \frac{1}{2} x^T D x \\ \text{subject to} \quad & \\ & Ax - b \leq 0 \\ & -x \leq 0. \end{aligned} \tag{8.21}$$

Here D is assumed to be positive semi-definite so that the objective function of (8.20) is a concave function (equivalently the objective function of (8.21) is a convex function). We now take the Lagrangian

$$L(x, \lambda, \mu) = -c^T x + \frac{1}{2} x^T D x + \lambda^T (Ax - b) - \mu^T x$$

giving

$$\nabla_x L(x, \lambda, \mu) = -c + Dx + A^T \lambda - \mu$$

and then its dual becomes

$$\begin{aligned} -\text{Max} \quad & L(x, \lambda, \mu) \\ \text{subject to} \quad & \\ & \nabla_x L = 0 \\ & \lambda \geq 0 \\ & \mu \geq 0. \end{aligned}$$

i.e.

$$\begin{aligned} -\text{Max} \quad & -c^T x + \tfrac{1}{2} x^T D x + \lambda^T (Ax - b) - \mu^T x \\ \text{subject to} \quad & \\ & -c^T + x^T D + \lambda^T A - \mu^T = 0 \\ & \lambda \geq 0 \\ & \mu \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} -\text{Max} \quad & -b^T\lambda + \tfrac{1}{2}x^TDx \\ \text{subject to} \quad & \\ & -c^T + x^TD + \lambda^TA - \mu^T = 0 \\ & \lambda \geq 0 \\ & \mu \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} \text{-Max} \quad & -b^T\lambda + \tfrac{1}{2}x^TDx \\ \text{subject to} \quad & \\ & -c + A^T\lambda + Dx = \mu \geq 0 \\ & \lambda \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} \text{-Max} \quad & -b^T\lambda + \tfrac{1}{2}x^TDx \\ \text{subject to} \quad & \\ & A^T\lambda + Dx \geq c \\ & \lambda \geq 0, \end{aligned}$$

i.e.

$$\begin{aligned} \text{Min} \quad & b^T\lambda - \tfrac{1}{2}x^TDx \\ \text{subject to} \quad & \\ & A^T\lambda + Dx \geq c \\ & \lambda \geq 0. \end{aligned} \tag{8.22}$$

problems (8.20) and (8.22) constitute the primal-dual pair of quadratic programming problems. In case $D = 0$, problems (8.20) and (8.22) become the linear programming primal-dual pair (8.18) and (8.19).

8.8 Summary and Additional Notes

- This Chapter gives a formal mathematical proof of the Karush-Kuhn-Tucker optimality conditions and uses the same to discuss the Lagrangian dual of the given nonlinear programming problem.
- Though there are several approaches of deriving the KKT optimality conditions, we have followed the *feasible directions* and *linearizing cone* approach of Zangwill [173].

- Another popular approach for studying duality in nonlinear programming is via the conjugate function approach due to Fenchel [55] and Rockafellar [136].
- Magnanti [108] showed that, for convex programming problems, the Fenchel and the Lagrangian duals are equivalent. We may refer to Avriel [6] for an elementary introduction of conjugate functions and related duality results.
- The Kuhn-Tucker optimality conditions were derived by Kuhn-Tucker in 1951. However, Karush originally derived these conditions in 1939 using calculus of variations; but his work could not get much attention as it was never published. But later when Karush's work came to notice, the Kuhn-Tucker optimality conditions were renamed as Karush-Kuhn-Tucker (KKT) optimality conditions.
- The sufficiency of KKT conditions has been extended to various other generalized convex functions, e.g. Mangasarian [109] for the case when the objective function is pseudo-convex and the constraint functions are quasi convex.
- The topic of optimality conditions in a nonsmooth setting is a very active field of research currently and we may refer to Bector et al. [13] for an elementary introduction of the same.

8.9 Exercises

8.1 *Use KKT theorem to find the value of β for which $(\bar{x}_1 = 1, \bar{x}_2 = 2)$ is optimal to the problem*

$$\begin{aligned} &Max && Z = 2x_1 + \beta x_2 \\ &\text{subject to} \\ & && x_1^2 + x_2^2 \leq 5 \\ & && x_1 - x_2 \leq 2\,. \end{aligned}$$

Verify your result graphically.

8.2 *Consider LPP*

$$\begin{aligned} &Max && Z = 4x_1 + 3x_2 \\ &\text{subject to} \\ & && x_1 + x_2 \leq 8 \\ & && 2x_1 + x_2 \leq 10 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

Use the KKT theorem to show that $(\bar{x}_1 = 2, \bar{x}_2 = 2)$ can not be an optimal solution of the given LPP.

8.3 *Consider the optimization problem*

$$\begin{aligned} &\textit{Max} && Z = x_2 \\ &\textit{subject to} \\ &&& |x_1| + |x_2| \leq 2 \\ &&& x_1^2 - x_2 \geq 0\,. \end{aligned}$$

1. *Solve the above problem graphically?*
2. *Do KKT conditions hold at (0,0)?*
3. *Can we conclude that (0,0) is optimal?*
4. *Is the given problem a convex programming problem?*

8.4 *Consider the NLP*

$$\begin{aligned} &\textit{Min} && Z = (x_1 - 4)^4 + (x_2 - 4)^2 \\ &\textit{subject to} \\ &&& x_1 + x_2 \leq 4 \\ &&& x_1 - x_2 \leq 2 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

Let $\hat{x} = (x_1 = 2, x_2 = 0)$ *be a feasible point. Determine* $D(\hat{x})$, $\overline{D}(\hat{x})$, $\mathfrak{D}(\hat{x})$, $Z_1(\hat{x})$ *and* $Z_2(\hat{x})$ *analytically and show them graphically as well. Do there exist KKT multipliers satisfying the KKT conditions at (2,0)?*

8.5 *Consider the NLP*

$$\begin{aligned} &\textit{Max} && Z = ln(1 + x_1) + x_2 \\ &\textit{subject to} \\ &&& 2x_1 + x_2 \leq 3 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

1. *Write the KKT conditions.*
2. *Given that the optimal solution of the above NLP lies on the line* $x_2 = 3$, *use the KKT conditions to find its optimal solution.*

8.6 *Consider the NLP*

$$\begin{aligned} &\textit{Min} && Z = x_1 + x_2 \\ &\textit{subject to} \\ &&& -x_1 x_2 + 1 \leq 0 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

1. *Write the KKT conditions of the above NLP.*
2. *Check if the KKT conditions hold at* $(x_1^* = 1, x_2^* = 1)$.
3. *Is* $(\bar{x}_1 = 1, \bar{x}_2 = 1)$ *optimal for the above NLP? Verify your answer graphically and give reasons for your answer in the light of the KKT theorem.*

8.7 *consider the constraints set* $x_1 + x_2 \leq 4$, $x_2 \geq 0$ *and* $x_1^2 + x_2^2 \leq 16$. *Let* $\hat{x} = (x_1 = 0, x_2 = 4)$. *Obtain* $D(\hat{x})$, *and* $\mathfrak{D}(\hat{x})$. *Does the basic constraint qualification holds at the point* $\hat{x}$*?*

8.8 *Consider the following NLP*

$$\begin{aligned} &\textit{Min} && Z = (x_1 - 4)^2 + (x_2 - 6)^2 \\ &\textit{subject to} \\ & && x_2 \geq x_1^2 \\ & && x_1 \leq 4. \end{aligned}$$

1. *Solve the above NLP graphically.*
2. *Write the KKT conditions.*
3. *Do the KKT conditions hold at the point (2,4)?.*
4. *Can we use the KKT conditions to declare that (2,4) is optimal? Give reasons for your answer.*

8.9 *Write the Wolfe dual (D) of following NLP (P)*

$$\begin{aligned} &\textit{Min} && Z = (x_1 - 4)^2 + (x_2 - 4)^2 \\ &\textit{subject to} \\ & && x_1 + x_2 \leq 4 \\ & && x_1 - x_2 \leq 2 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

Solve (P) and (D) and hence verify the strong duality theorem.

8.10 *Write the Wolfe dual (D) of following NLP (P)*

$$\begin{aligned} &\textit{Max} && Z = 2x_1 + 4x_2 \\ &\textit{subject to} \\ & && x_1^2 + x_2^2 \leq 5 \\ & && x_1 - x_2 \leq 2. \end{aligned}$$

It is claimed that the optimal value of (D) is 10. Do you agree with the claim? Justify your answer using duality theory.

8.11 *Let* $x_1, x_2 \in \mathbf{R}$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, $\lambda_3 \geq 0$, *and* $L(x_1, x_2, \lambda_1, \lambda_2, \lambda_3) = (4x_1^2 + x_2^2 - 2x_1x_2) + \lambda_1(x_1 + x_2 - 1) - \lambda_2 x_1 - \lambda_3 x_2$. *Describe explicitly the following problems*

$\underset{(x_1,x_2)}{Min}\left(\underset{(\lambda_1,\lambda_2,\lambda_3)}{Max} L(x_1, x_2, \lambda_1, \lambda_2, \lambda_3)\right)$ *and* $\underset{(\lambda_1,\lambda_2,\lambda_3)}{Max}\left(\underset{(x_1,x_2)}{Min} L(x_1, x_2, \lambda_1, \lambda_2, \lambda_3)\right)$. *It is claimed that both of these problems have optimal solutions and equal optimal values. Justify your answer and obtain the common optimal value.*

8.12 *Consider the following NLP (P)*

$$\begin{aligned} &Max && Z = x_1 \\ &\text{subject to} \\ &&& x_1^2 + x_2^2 \leq 1 \\ &&& (x_1 - 1)^2 \leq x_2. \end{aligned}$$

1. *Write the KKT conditions and check if these are satisfied at (1,0).*
2. *Solve the above NLP graphically and hence obtain its optimal solution* $(\bar{x}_1, \bar{x}_2)$.
3. *Do there exist KKT multipliers at the point* $(\bar{x}_1, \bar{x}_2)$*? Give reasons for your answer.*
4. *Is the above a convex programming problem? Give reasons for your answer?*
5. *Write the Wolfe dual (D) of (P). Can your guarantee that (P) and (D) will have equal optimal values.*

9
Unconstrained Optimization Problems

9.1 Introduction

An optimization problem in which the decision vector x is allowed to take any value in $\mathbf{R}^n$ is called an *unconstrained optimization problem*, written in short as (UMP). This chapter is devoted to the study of certain standard algorithms for unconstrained minimization of functions of one variable as well as functions of several variables. Although most real life optimization problems are constrained optimization problems, the study of unconstrained optimization problems is important, mainly because certain efficient techniques for solving constrained optimization problems use our knowledge of solving unconstrained optimization problems.

9.2 Basic Scheme and Certain Desirable Properties

We consider the unconstrained minimization problem

$$\underset{x \in \mathbf{R}^n}{Min} \quad f(x) \tag{9.1}$$

and aim to develop algorithms for solving the same. Ideally we shall like to get a global min point of (9.1), i.e. to get a point $\bar{x} \in \mathbf{R}^n$ such that $f(\bar{x}) \leq f(x)$ for all $x \in \mathbf{R}^n$. But unfortunately this seems to be rather difficult for a general function f. Even finding a local min point, i.e. a point $\bar{x}$ such that there exists a neighbourhood $N_\delta(\bar{x})$ with $f(\bar{x}) \leq f(x)$ for all $x \in N_\delta(\bar{x})$, is also not always possible.

In view of the above, we most often decide to identify a set $\Omega \subseteq \mathbf{R}^n$, called the *solution set*, such that under certain additional conditions on the function f (e.g. convexity) any point of Ω becomes a global min point of problem (9.1).

So far we have not put any smoothness condition on the function f. Let us now assume that the function f is continuously differentiable. Sometimes we may have also to assume that f is twice continuously differentiable but this condition we shall mention specifically if needed.

Definition 9.2.1 (Solution Set). *Let $f : \mathbf{R}^n \longrightarrow \mathbf{R}$ be continuously differentiable. Then the set*

$$\Omega = \{ x \in \mathbf{R}^n : \nabla f(x) = 0 \}$$

is called the solution set of (9.1).

Remark 9.2.1 *Let $f : \mathbf{R}^n \longrightarrow \mathbf{R}$ be differentiable convex function and $\bar{x}$ be a point of the solution set. Then $\bar{x}$ is a global min point of problem (9.1).*

We now describe a common *basic scheme* of the form

$$x^{(k+1)} = x^{(k)} + \overline{\alpha_k}\, d^{(k)}$$

for solving the unconstrained minimization problem (9.1) where $x^{(k)}$ is the current solution, $d^{(k)}$ is the *direction of movement* from $x^{(k)}$ and $\overline{\alpha_k} > 0$ (called the *step size*) is the distance upto which we move in the direction $d^{(k)}$ from the current point $x^{(k)}$. The obvious question now is how to find the direction for movement $d^{(k)}$ and how to determine the step size $\overline{\alpha_k}$? Various algorithms for solving problem (9.1) have been devised to determine the direction $d^{(k)}$ and the step size $\overline{\alpha_k}$ so that the sequence of iterates $\{x^{(k)}\}$ converges to a point $\bar{x}$ of the solution set Ω in an 'efficient' manner.

Before we present any specific algorithm for solving problem (9.1), we list certain desirable properties which we ideally expect the given algorithm to possess.

Definition 9.2.2 (Descent Property). *An algorithm for solving the unconstrained minimization problem (9.1) is said to have the descent property if the objective function value decreases as we go through the sequence $\{x^{(k)}\}$, i.e.*

$$f(x^{(k+1)}) < f(x^{(k)}) \text{ for all } k.$$

In other words, for the algorithm to possess the descent property, the objective function should decrease as we proceed.

Definition 9.2.3 (Quadratic Termination Property). *An algorithm for the unconstrained minimization problem (9.1) is said to have quadratic termination property (q.t.p) if the minimum of a positive definite quadratic form in n variables is reached in at most n iterations.*

The motivation for defining q.t.p stems from the fact that near a local min point, the function behaves like a strictly convex function(like a parabola in $\mathbf{R}$ or a positive definite quadratic form in $\mathbf{R}^n$) and therefore if the algorithm behaves well on such a function, it will 'hopefully' do well on the other functions as well.

Definition 9.2.4 (Globally Convergent). *An algorithm for the unconstrained minimization problem (9.1) is said to be globally convergent if starting from any point $x^{(0)} \in \mathbf{R}^n$, the sequence $\{x^{(k)}\}$ always converges to a point of the solution set Ω.*

The property of 'global convergence' guarantees that we can start the algorithm from any arbitrary point $x^{(0)} \in \mathbf{R}^n$. Thus no matter from which point we start, we are guaranteed to generate a sequence $\{x^{(k)}\}$ that converges to a point of the solution set. Here it must be noted that two different starting points will, in general, generate two different sequences of iterates but both converging to a point of solution set.

Definition 9.2.5 (Order of Convergence). *Let the sequence $\{x^{(k)}\}$ converge to a point $\bar{x}$ and let $x^{(k)} \neq \bar{x}$ for sufficiently large k. The quantity $\|x^{(k)} - \bar{x}\|$ is called the error of the kth iterate $x^{(k)}$. Suppose that there exists p and $0 < a < \infty$ such that*

$$\lim_{k\to\infty} \frac{\|x^{(k+1)} - \bar{x}\|}{\|x^{(k)} - \bar{x}\|^p} = a \quad (0 < a < \infty)$$

then p is called the order of convergence of the sequence $\{x^{(k)}\}$. Thus $\|x^{(k+1)} - \bar{x}\| = a\|x^{(k)} - \bar{x}\|^p$ asymptotically.

If $p = 1$, the sequence $\{x^{(k)}\}$ is said to have *linear convergence rate* and for $p = 2$, it is said to have *quadratic convergence rate*. In case $p = 1$ but $a = 0$, then the sequence $\{x^{(k)}\}$ is said to have *super linear convergence rate*.

The order of convergence or convergence rate is an important concept as it tells us how the 'tail' of the sequence $\{x^{(k)}\}$ behaves. Larger values of p will imply faster convergence. Most of the algorithms generally do very well for the first few iterations but become very slow near the optimal solution. But if p is large then there will be significant improvement in the objective function value even near the actual solution.

We now discuss *line search methods* or *one dimensional search methods* which apart from being useful in themselves are used to determine the step size $\overline{\alpha_k}$ in the basic scheme

$$x^{(k+1)} = x^{(k)} + \overline{\alpha_k}\, d^{(k)} .$$

9.3 Line Search Methods for Unimodal Functions

These methods are used to locate min (or max) point of a *unimodal* function of one variable over a interval, i.e. we are given a unimodal function $f : [a,b] \longrightarrow \mathbf{R}$ and we wish to locate the minimizing point x_{min} such that

$$f(x_{min}) = \underset{a\le x\le b}{Min}\, f(x) . \tag{9.2}$$

Definition 9.3.1 (Unimodal Min Function). *The function $f : [a,b] \longrightarrow \mathbf{R}$ is said to be a unimodal (to be specific unimodal min) function if it has only one mode i.e. it has a single relative min, i.e. $\exists\ a \le \alpha \le b$ such that*
(i) f is strictly decreasing ($\downarrow$) in $[a,\alpha)$.
(ii) f is strictly increasing ($\uparrow$) in $[\alpha,b]$.

Similarly we define a *unimodal max function*. Note that a unimodal function may not be differentiable - in fact it may not even be continuous, as depicted in Fig 9.1.

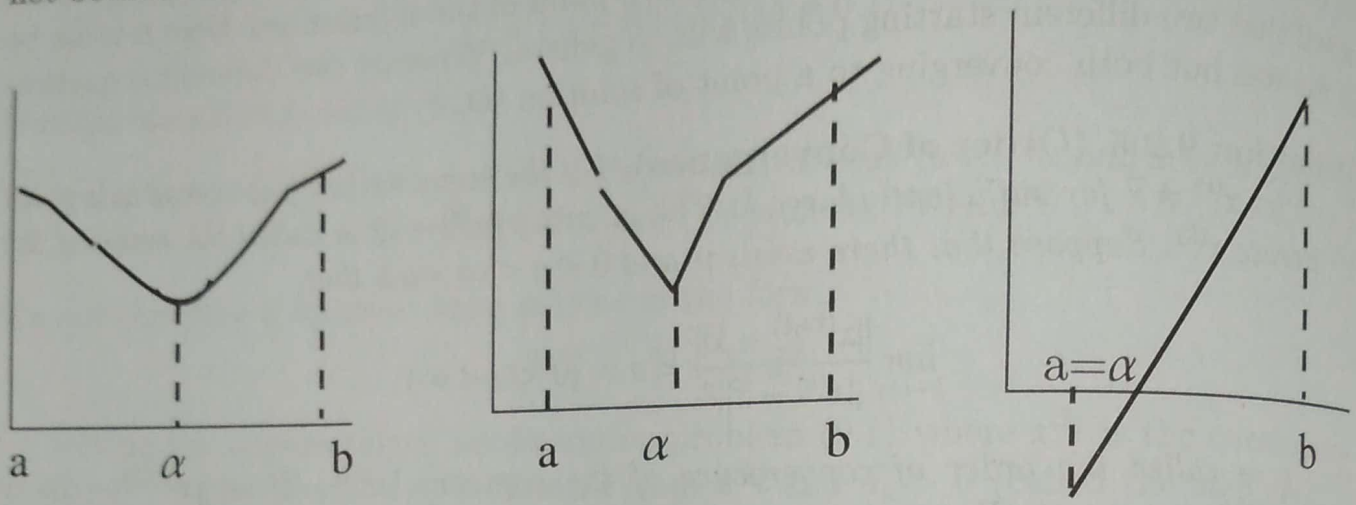


Fig. 9.1.

Basic Strategy

Let us choose two distinct points (say) x_1 and x_2 in $[a,b]$ i.e. $a \leq x_1 < x_2 \leq b$. Let x_{min} denote the point giving the minimum value of $f(x)$ over $[a,b]$. Since f is unimodal min function, it is clear that

(i) $f(x_1) < f(x_2) \Rightarrow x_{min} \in [a, x_2]$
(ii) $f(x_1) > f(x_2) \Rightarrow x_{min} \in [x_1, b]$
(iii) $f(x_1) = f(x_2) \Rightarrow x_{min} \in [x_1, x_2]$.

Let us agree to include case (iii) in case (i) (as case (iii) is not that likely). Then we observe that by evaluating the value of the function at two distinct points, the initial search length $(b-a)$(i.e. the interval $[a,\ b]$) has been reduced to (x_2-a) (i.e. the interval $[a,\ x_2]$) or $(b-x_1)$ (i.e. the interval $[x_1,\ b]$). The interval $[a,b]$ is called the initial *interval of uncertainty*. The basic question in line search methods of this type is to develop strategies for choosing points $x_1, x_2, ..., x_N$ such that $a \leq x_1 < x_2 < ... < x_{N-1} < x_N \leq b$ so that we can determine the smallest possible interval of uncertainty in which the minimizing point must lie. Note that we do not need explicit knowledge of the function f as long as it is possible to get its value at a specified point. Further we do not make any continuity/diffrentiability assumption on f. But then we have to assume that f is a unimodal min function.

There could be various search methods depending upon how the N trial points $x_1, x_2, ..., x_N$ have been chosen. It should be noted that because every function evaluation may be costly in time, we do not wish to evaluate the function f at too many points.

We shall be studying the following two search methods for the case of unimodal function. These are

(i) The Golden Section Rule.
(ii) The Fibonacci Search Method.

In discussing the aforesaid methods we shall make use of the following notations

$x_{L,k}$ = lower limit of the search interval at the kth iteration (so $x_{L,1}$ = a)
$x_{U,k}$ = upper limit of the search interval at the kth iteration (so $x_{U,1}$ = b)
$x_{p,k}$ = 1st trial point at the kth iteration
$x_{q,k}$ = 2nd trial point at the kth iteration
$E_{p,k} = f(x_{p,k})$ value of the function at the 1st trial point
$E_{q,k} = f(x_{q,k})$ value of the function at the 2nd trial point
$I_k = x_{U,k} - x_{L,k}$ = length of the search interval at the kth iteration (so I_1 = (b - a))
I_k^L = length of the left part of the search interval I_k

I_k^R = length of the right part of the search interval I_k.

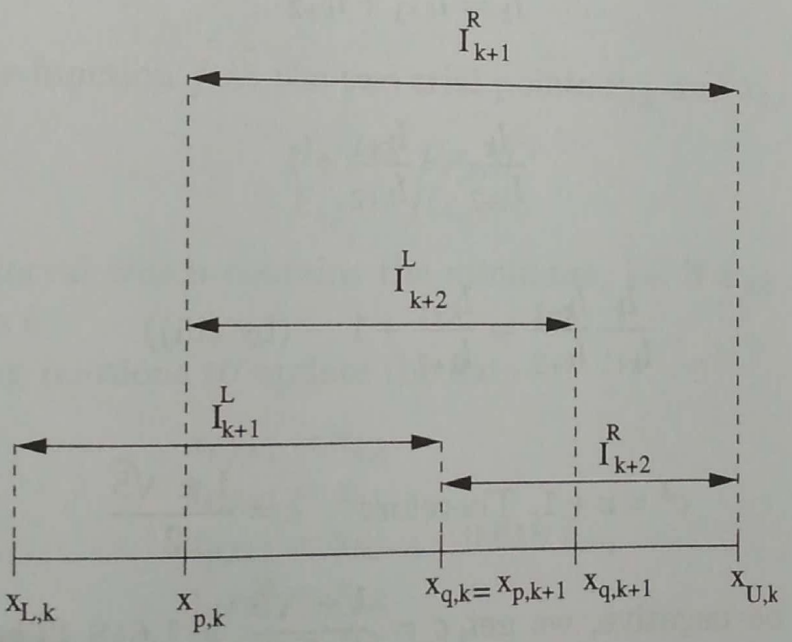


Fig. 9.2.

The Golden Section Rule

Here the two trial points $x_{p,k}$ and $x_{q,k}$ are chosen as per the following criteria
(i) $I_k^L = I_k^R$ for all k , i.e. no undue advantage is given to left or right part.
(ii) $x_{p,k+1} = x_{q,k}$ and $x_{q,k+1}$ is computed afresh for the interval I_{k+1}^R (or $x_{q,k+1} = x_{p,k}$ and $x_{p,k+1}$ is computed afresh for the interval I_{k+1}^L), i.e. only one new trial point is computed

at the kth iteration, the other trial point is obtained from the previous iteration. In view of this we have

$$I_k = I_{k+1} + I_{k+2} \quad \text{for all } k.$$

(see Fig 9.2.)

(iii) $\dfrac{I_k}{I_{k+1}} = \dfrac{I_{k+1}}{I_{k+2}} = \text{c (constant) for all } k.$

This is called the *golden section criterion*. In general, a point w is said to divide the interval $[u, v]$ according to the golden section criterion if we have

$$\frac{\text{length of the whole interval}}{\text{length of the bigger interval}} = \frac{\text{length of the bigger interval}}{\text{length of the smaller interval}}$$

i.e. $$\frac{(v-u)}{(w-u)} = \frac{(w-u)}{(v-w)}.$$

Thus we wish to choose trial points $x_{p,k}$ and $x_{q,k}$ according to the above listed criterion (i), (ii) and (iii). From (ii) we have

$$I_k = I_{k+1} + I_{k+2}$$

i.e.

$$\frac{I_k}{I_{k+2}} = \frac{I_{k+1}}{I_{k+2}} + 1$$

i.e.

$$\frac{I_k}{I_{k+1}} \frac{I_{k+1}}{I_{k+2}} = \frac{I_{k+1}}{I_{k+2}} + 1 \quad \text{(by (iii))}$$

i.e.

$$c^2 = c + 1. \text{ Therefore } \quad c = \frac{1 \pm \sqrt{5}}{2}$$

Since c can not be negative, we get $c = \dfrac{1+\sqrt{5}}{2} = 1.618$ (The number $c = 1.618$ is called the *golden section ratio* and it has a long history - something to do with our aesthetic value). Note that $\dfrac{1}{1.618} = 0.618$.

Let us now imagine that the initial search interval is [0, 1]. Then due to the above arguments

$$\frac{1 - x_{p,1}}{x_{p,1}} = \frac{1}{1 - x_{p,1}} = 1.618.$$

Therefore
$$x_{p,1} = 0.382$$
$$x_{q,1} = 1 - 0.382 = 0.618 .$$

For a general interval $[a, b]$, we can then have

$$x_{p,1} = b - 0.618\,(b - a)$$
$$x_{q,1} = a + 0.618\,(b - a) .$$

Thus in general,

$$x_{p,k} = x_{U,k} - 0.618\,I_k$$
$$x_{q,k} = x_{L,k} + 0.618\,I_k .$$

The stepwise description of the method is now as follows

Step 1 Input data- $x_{L,1}$, $x_{U,1}$, ϵ, f (Here $\epsilon > 0$ is the tolerance to be prescribed by the user).

Step 2 Compute the first two trial points $x_{p,1}$ and $x_{q,1}$, where

$$x_{p,1} = x_{U,1} - 0.618\,(x_{U,1} - x_{L,1})$$
$$x_{q,1} = x_{L,1} + 0.618\,(x_{U,1} - x_{L,1}).$$

Set $k = 1$.

Step 3 Evaluate the function f at the two trial points $x_{p,k}$ and $x_{q,k}$. Let

$$E_{p,k} = f(x_{p,k})$$
$$E_{q,k} = f(x_{q,k}) .$$

Step 4 Test the interval which contains the minimum, i.e. if $E_{p,k} \leq E_{q,k}$, go to Step 5 otherwise go to Step 6.

Step 5 Use following relations to update the data

$$x_{L,k+1} = x_{L,k}$$
$$x_{U,k+1} = x_{q,k}$$
$$x_{p,k+1} = x_{U,k+1} - 0.618\,I_{k+1}$$
$$x_{q,k+1} = x_{p,k}$$
$$E_{p,k+1} = f(x_{p,k+1})$$
$$E_{q,k+1} = f(x_{q,k+1}) = E_{p,k} .$$

Step 6 Use the following relations to update the data

$$x_{L,k+1} = x_{p,k}$$
$$x_{U,k+1} = x_{U,k}$$
$$x_{p,k+1} = x_{q,k}$$
$$x_{q,k+1} = x_{L,k} + 0.618\,I_{k+1}$$
$$E_{p,k+1} = f(x_{p,k+1}) = E_{q,k}$$
$$E_{q,k+1} = f(x_{q,k+1}) .$$

Step 7 Test for the end of optimization , i.e. if $I_k < \epsilon$, go to Step 8, otherwise set k = (k+1) and go to Step 4.
Step 8 Output $x_{L,n}$, $x_{U,n}$. Then $x_{min} \in (x_{L,n}, x_{U,n})$ and $E_{min} \leq Min(E_{p,n-1}, E_{q,n-1})$.

Remark 9.3.1 *It can be noted that for the golden section rule,* $\frac{I_n}{I_1} = \left(\frac{1}{1.618}\right)^{n-1} \approx (0.618)^{n-1}$. *Thus knowing* I_1 *(the length of the initial search interval) and* I_n *(the length of the final search interval as desired by the user) we can find out the number of iterations as well as the number of points at which the function is to be evaluated. Note that if we are going upto the* 7th *iteration (i.e. getting* $x_{L,7}$ *and* $x_{U,7}$*) then we shall be having* $n = 7$ *functional evaluations, namely, at* $x_{p,1}$, $x_{q,1}$, *and one each at* 2nd, 3rd, 4th, 5th *and* 6th *iteration. To stop at the* 7th *iteration we shall be computing* $x_{p,6}$ *and* $x_{q,6}$ *and* $E_{min} \leq Min\ (x_{p,6}, x_{q,6})$.

Thus N functional evaluations will mean stopping at the Nth *iteration and computing points upto* $x_{p,N-1}$ *and* $x_{q,N-1}$. *Also* $E_{min} \leq Min\ (x_{p,N-1}, x_{q,N-1})$ *and there are only* $(N-1)$ *interval reductions.*

Example 9.3.1 *Find min* x^2 *over [-5, 15] by the golden section rule. Take* $\epsilon = 1.5$.

Solution Following the steps of the golden section rule we get

k	$x_{L,k}$	$x_{U,k}$	$x_{p,k}$	$x_{q,k}$	$E_{p,k}$	$E_{q,k}$	L/R
1	- 5.0	15.00	2.64	7.36	6.96	54.1	L
			↘		↘		
2	-5.0	7.36	-0.2	2.64	0.077	6.96	L
			↘		↘		
3	-5.0	2.64	-2.08	-0.27	4.33	0.077	R
			↙		↙		
4	-2.08	2.64	-0.2	0.84	0.077	0.71	L
			↘		↘		
5	-2.08	0.84	-0.96	-0.27	0.92	0.077	R
			↙		↙		
6	-0.96	0.84	-0.27	0.15	0.077	0.023	R
7	-0.27	0.84					

Here $I_7 = 1.11 < 1.5(= \epsilon)$, so we stop. Thus $x_{min} \in [-0.27, 0.84]$ and $E_{min} \leq$ Min (0.077, 0.023) = 0.023.

Note that this example is just for the sake of illustration. Normally the function will not be so well behaved and ϵ (tolerance) will be taken much smaller e.g. $\epsilon = 0.001$ etc.

The Fibonacci Search Method

We continue our discussion with regard to problem (9.2) and present another approach which is based on the Fibonacci sequence.

Definition 9.3.2 (Fibonacci Sequence). *Let* $F_0 = 1$, $F_1 = 1$ *and* $F_i = F_{i-1} + F_{i-2}$ $(i \geq 2)$. *Then* $\{F_n\}$ *is called the sequence of Fibonacci numbers or in short, the Fibonacci sequence. Thus the Fibonacci sequence is* $\{1, 1, 2, 3, 5, 8, 13, 21, \ldots\}$ *where* $F_0 = 1$, $F_1 = 1$, $F_2 = 2$, $F_3 = 3$, $F_4 = 5$, $F_5 = 8$, $F_6 = 8$, $F_7 = 21 \ldots$ *and so on.*

Now we describe the Fibonacci search scheme. Here the first two conditions are exactly same as in the golden section rule, but the third condition changes. Thus we have

(i) $I_k^L = I_k^R$ for all k
(ii) $I_k = I_{k+1} + I_{k+2}$
(iii) $\dfrac{I_k}{I_{k+1}} = \dfrac{F_{n-k+1}}{F_{n-k}}$, where n denotes the number of iterations to be performed.

We now give a justification of the condition (iii) given above. If we wish to stop at the nth iteration, then

$$\begin{aligned}
I_{n+1} &\simeq I_n && = 1\ I_n \\
I_n &= I_n && = 1\ I_n \\
I_{n-1} &= I_n + I_{n+1} && = 2\ I_n \\
I_{n-2} &= I_n + I_{n-1} && = 3\ I_n \\
I_{n-3} &= 5\ I_n \\
I_{n-4} &= 8\ I_n \\
&\vdots \\
I_1 &= F_n I_n.
\end{aligned}$$

Thus we can see at once that the role of the Fibonacci numbers, because $I_1/I_n = F_n$, or in general, $\dfrac{I_k}{I_{k+1}} = \dfrac{F_{n-k+1}}{F_{n-k}}$. Hence the only difference between the golden section rule and the Fibonacci search method is the requirement (iii). In the golden section rule $\dfrac{I_k}{I_{k+1}} = c = 1.618$ for all k and all n. Here, in the Fibonacci search method this ratio depends on both the current iteration k and also the total number of iterations n to be performed. Therefore instead of 0.618 (i.e. $\dfrac{1}{c}$) we shall be using the number $\dfrac{F_{n-k}}{F_{n-k+1}}$ for determining the points $x_{p,k}$ and $x_{q,k}$.

The stepwise description of the Fibonacci search method is as follows

Step 1 Input data: $x_{L,1}$, $x_{U,1}$, n, f.
Step 2 Compute Fibonacci numbers F_n and F_{n-1} and find

$$x_{p,1} = x_{U,1} - \frac{F_{n-1}}{F_n}(x_{U,1} - x_{L,1})$$
$$x_{q,1} = x_{L,1} + \frac{F_{n-1}}{F_n}(x_{U,1} - x_{L,1})$$

Set $k = 1$.

Step 3 Evaluate

$$E_{p,k} = f(x_{p,k})$$
$$E_{q,k} = f(x_{q,k})$$

and test the interval which contains the minimum. If $E_{p,k} \leq E_{q,k}$, go to Step 4, otherwise go to Step 5.

Step 4 Use the following relations to update the data

$$x_{L,k+1} = x_{L,k}$$
$$x_{U,k+1} = x_{q,k}$$
$$x_{p,k+1} = x_{U,k+1} - \frac{F_{n-k}}{F_{n-k+1}} I_{k+1}$$
$$x_{q,k+1} = x_{p,k}$$
$$E_{p,k+1} = f(x_{p,k+1})$$
$$E_{q,k+1} = f(x_{q,k+1}) = E_{p,k} \ .$$

Step 5 Use the following relations to update the data

$$x_{L,k+1} = x_{p,k}$$
$$x_{U,k+1} = x_{U,k}$$
$$x_{p,k+1} = x_{q,k+1}$$
$$x_{q,k+1} = x_{L,k} + \frac{F_{n-k}}{F_{n-k+1}} I_{k+1}$$
$$E_{p,k+1} = f(x_{p,k+1}) = E_{q,k}$$
$$E_{q,k+1} = f(x_{q,k+1}) \ .$$

Step 6 If $k \leq (n-2)$, then set $k = k+1$ and go to step 3. If $k = (n-1)$ then go to Step 7.

Step 7 As $F_0 = F_1 = 1$, it can be seen that at the $(n-1)^{th}$ iteration, the two trial points $x_{p,n-1}$ and $x_{q,n-1}$ will come out to be the same , i.e. $x_{p,n-1} = x_{q,n-1}$. To make them distinct we have to add / subtract a fixed positive number ϵ (say 0.01) to one of these. For this we look at the $(n-2)^{th}$ iteration. If there (i.e. at the $(n-2)^{th}$ iteration) we are searching in the left then ϵ is subtracted from $x_{p,n-1}$ otherwise it is added to $x_{q,n-1}$. We then determine new $x_{p,n-1}$ and $x_{q,n-1}$ and hence $x_{L,n}$ and $x_{U,n}$. Then

$$x_{min} \in (x_{L,n}, x_{U,n})$$
and
$$f_{min} \leq \text{Min}(E_{p,n-1}, E_{q,n-1})$$

Remark 9.3.2 *As* $\frac{I_n}{I_1} = \frac{1}{F_n}$, *so given* I_1 *and* I_n, *the value of* n *can be computed in advance.*

Example 9.3.2 *Taking $n = 7$, find min x^2 over [-5,15] by the Fibonacci search method.*

Solution Following the steps of the Fibonacci search method we get

k	F_{n-k}/F_{n-k+1}	$x_{L,k}$	$x_{U,k}$	$x_{p,k}$	$x_{q,k}$	$E_{p,k}$	$E_{q,k}$	L/R
1	13/21	-5.0	15.00	2.64	7.38	6.88	54.6	L
				↘		↘		
2	8/13	-5.0	7.38	-0.24	2.62	0.058	6.88	L
				↘		↘		
3	5/8	-5.0	2.62	-2.14	-0.24	4.67	0.058	R
				↙		↙		
4	3/5	-2.14	2.62	-0.24	0.72	0.058	0.52	L
				↘		↘		
5	2/3	-2.14	0.72	-1.2	-0.24	1.44	0.058	R
				↙		↙		
6	1/2	- 1.2	0.72	-0.24	-0.24+0.01	0.058	0.053	R
7	1	-0.24	0.72					

From this table we observe that $x_{min} \in [-0.24, 0.72]$ and $f_{min} \leq 0.053$.

In the given example, $I_1 = 20$ and therefore if we take $\epsilon = 1.5$, then $I_n = 1.5$. This gives $\frac{I_1}{I_n} = \frac{20}{1.5} = 13.3$ which from the sequence of Fibonacci numbers determines $n = 7$ as $F_6 = 13$ and $F_7 = 21$. Also as explained in Step 7, the two trial points at the $(n-1)^{th}$ iteration, i.e. $x_{p,6}$ and $x_{q,6}$ both becomes equal to 0.24. Therefore we look at the $(n-2)^{th}$ iteration, i.e. the 5th iteration in our example. As there we are seeking the right part, we add 0.01 to $x_{q,6}$ to get new $x_{q,6}$ and then continue as explained.

Relation Between the Fibonacci Search Method and the Golden Section Rule

For the constant c appearing in the golden section rule we have $c = 1 + 1/c = 1.618$. Also it can be proved that $F_n \simeq \frac{c^{n+1}}{\sqrt{5}}$ (for large n). We shall now like to compare these two methods for the same number of function evaluations.

Let R_F and R_{GS} denote the reduction factors for the Fibonacci search method and the golden section rule respectively. Then

$$R_F = \frac{I_n}{I_1} = \frac{1}{F_n} = \frac{\sqrt{5}}{c^{n+1}}$$

and

$$R_{GS} = \frac{I_n}{I_1} = \left(\frac{1}{c}\right)^{n-1} = \frac{1}{c^{n-1}}.$$

Therefore

$$\frac{R_{GS}}{R_F} = \frac{1}{c^{n-1}} \times \frac{c^{n+1}}{\sqrt{5}} = \frac{c^2}{\sqrt{5}} \simeq 1.17 .$$

Thus for the same number of function evaluations, the final search interval for the Fibonacci search method will be 17% smaller than the one obtained by the golden section rule .

9.4 The Steepest Descent Method

The steepest descent method is one of the oldest gradient based methods for solving an unconstrained optimization problem. This method is extremely simple to implement and therefore has been used widely in various applications. The only drawback of the steepest descent method is its slow convergence (as its order of convergence is one). But this has motivated researchers to develop more advanced algorithms by modifying the basic descent strategy of the steepest descent method so that these algorithms have superior convergence properties. We shall certainly discuss some of these algorithms in the subsequent sections.

Let us consider the unconstrained optimization problem (UMP)

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \; f(x) \tag{9.3}$$

where f has continuous first order partial derivatives on $\mathbf{R}^n$. Then the basic scheme of the steepest descent method can be described as follows

$$x^{(k+1)} = x^{(k)} + \overline{\alpha_k}\, d^{(k)}$$

where $d^{(k)} = -\nabla f(x^{(k)})/\|\nabla f(x^{(k)})\|$, is the direction of movement and the step size $\overline{\alpha_k} \geq 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k \geq 0}{Min}\; h(\alpha_k).$$

Here the function $h(\alpha_k)$ is given by $h(\alpha_k) = f(x^{(k)} + \alpha_k\, d^{(k)})$, and we stop when $\|\nabla f(x^{(k)})\| < \epsilon$.

Thus a stepwise description of the steepest descent method could be

Step 1 Choose $x^{(0)} \in \mathbf{R}^n$ and tolerance $\epsilon > 0$. Set k = 0.

Step 2 Compute $\nabla f(x^{(k)})$ and $d^{(k)} = -\nabla f(x^{(k)})/\|\nabla f(x^{(k)})\|$.

Step 3 Evaluate $x^{(k+1)} = x^{(k)} + \overline{\alpha_k}\, d^{(k)}$ where $\overline{\alpha_k} > 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k \geq 0}{Min}\; h(\alpha_k) ,$$

Here,

$$h(\alpha_k) = f(x^{(k)} + \alpha_k\, d^{(k)}) .$$

The Steps 2 and 3 above are repeated till $\|\nabla f(x^{(k)})\| < \epsilon$. In that case $x^{(k)}$ becomes a point of the solution set and therefore, if f is a convex function then it becomes an optimal solution of the unconstrained minimization problem (9.3).

The natural question here is to justify the particular choice of the direction $d^{(k)}$ at the current point $x^{(k)}$, i.e. why should we choose $d^k = -\nabla f(x^{(k)})/\|\nabla f(x^{(k)})\|$? For this let us recall the definition of the directional derivative of f at the point x in the direction d as

$$\left.\frac{\partial f}{\partial d}\right|_x = \lim_{\alpha \to 0^+} \frac{f(x+\alpha d) - f(x)}{\alpha}.$$

and note that for the case when f has continuous first order partial derivatives, we have

$$\left.\frac{\partial f}{\partial d}\right|_x = \frac{d^T}{\|d\|} \nabla f(x).$$

Therefore for $d^{(k)} = -\dfrac{\nabla f(x^{(k)})}{\|\nabla f(x^{(k)})\|}$, we get

$$\left.\frac{\partial f}{\partial d^{(k)}}\right|_{x=x^{(k)}} = -\frac{\|\nabla f(x^{(k)})\|^2}{\|\nabla f(x^{(k)})\|} = -\|\nabla f(x^{(k)})\| < 0.$$

As for $d^{(k)} = -\dfrac{\nabla f(x^{(k)})}{\|\nabla f(x^{(k)})\|}$, $\left.\dfrac{\partial f}{\partial d^{(k)}}\right|_{x=x^{(k)}} < 0$, it makes sense to move in the direction $d^{(k)}$ if we wish to minimize the function $f(x)$. In fact the directional derivative of f at x in the direction d is least when $d = -\dfrac{\nabla f(x)}{\|\nabla f(x)\|}$ because for the given x the optimization problem

$$\begin{aligned} \text{Min} \quad & d^T \nabla f(x) \\ \text{subject to} \quad & \\ & \|d\| \le 1. \end{aligned} \tag{9.4}$$

has the optimal solution $\bar{d} = -\dfrac{\nabla f(x)}{\|\nabla f(x)\|}$ with the optimal value $-\|\nabla f(x)\| < 0$.

Using any of the above arguments we must get convinced that if we are to minimize (locally) a function f from the current point $x^{(k)}$, then we must move in the direction of negative gradient and therefore we choose $d^{(k)}$ as the unit vector in that direction, i.e.

$$d^{(k)} = -\frac{\nabla f(x^{(k)})}{\|\nabla f(x^{(k)})\|}.$$

Next point to understand is the procedure for choosing the step size $\bar{\alpha}_k$, where the direction of movement $d^{(k)}$ is known. For this we consider all points $x^{(k)} + \alpha_k d^{(k)}$, $\alpha_k > 0$,

i.e. all points on the ray originating at $x^{(k)}$ and in the direction $d^{(k)}$. Here it should be emphasized that as f is nonlinear, $\nabla f(x^{(k)})$ is a function of $x^{(k)}$, i.e. the function f is NOT going to decrease for all time to come, in the direction $d^{(k)}$, i.e. there must exist some $\overline{\alpha_k} > 0$ beyond which the function will not decrease in the direction $d^{(k)}$. A common sense argument suggests that the easiest (not necessarily the best) way to choose $\overline{\alpha_k}$ could be to consider all values $x^{(k)} + \alpha_k\, d^{(k)}$, i.e. consider the function $h(\alpha_k) = f(x^{(k)} + \alpha_k\, d^{(k)})$ and choose $\overline{\alpha_k} > 0$ for which $h(\alpha_k)$ is minimum. Here we should note that $h(\alpha_k)$ is a function of one variable, namely α_k, only and so this minimization can be done numerically as well. Therefore $\overline{\alpha_k} > 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\; h(\alpha_k)$$

as described in the basic scheme of the steepest descent method.

We should now go back and try to verify which of the so called 'desirable properties' the steepest descent method possesses. Without proving any of these we shall below state the given result (and this we shall follow for all algorithms discussed in this chapter) for the steepest descent method.

Result 9.4.1 *The steepest descent algorithm*
(i) has descent property
(ii) does not have quadratic termination property
(iii) is globally convergent and
(iv) has order of convergence $p = 1$.

Therefore if we are using the steepest descent method then we can start from any point $x^{(0)}$ and as we proceed, the objective function value will decrease, but the algorithm may take lot many iterations near the optimal solution. Also it may, in general, take more than n iterations to minimize a positive definite quadratic form of n variables.

We now illustrate the working of the steepest descent method for the example given below.

Example 9.4.1 *Use the steepest descent method to minimize* $f(x_1, x_2) = 3x_1^2 - 4x_1x_2 + 2x_2^2 + 4x_1 + 6$ *over* $(x_1, x_2) \in \mathbf{R}^2$.

Solution The given function is a convex function and so the steepest descent method will give a global optimal solution. Starting from $x^{(0)} = (0, 0)^T$ and following Steps 1-3 described above, we get

k	$x^{(k)}$	$\nabla f(x^{(k)})$	$d^{(k)}$	$\overline{\alpha_k}$
1	$(0,0)^T$	$(4,0)^T$	$(-1,0)^T$	2/3
2	$(-2/3,0)^T$	$(0,8/3)^T$	$(0,-1)^T$	2/3
3	$(-2/3,-2/3)^T$	$(8/3,0)^T$	$(-1,0)^T$	1/6
4	$(-10/9,-2/3)^T$	$(0,16/9)^T$	$(0,-1)^T$	1/4
5	$(-38/27,-10/9)^T$	$(16/9,0)^T$	$(-1,0)^T$	1/6
⋮	⋮	⋮	⋮	⋮

Remark 9.4.1 *The function $f(x_1,x_2)$ of Example (9.4.1) is a positive definite quadratic form in two variables but its optimal solution has not been obtained in atmost two iterations. This illustrates that the method of steepest descent does not possess quadratic termination property.*

Remark 9.4.2 *Looking at the table for Example (9.4.1) we observe that the directions $d^{(k)}$ are repeated alternately $(-1,0)^T,(0,-1)^T,(-1,0)^T,(0,-1)^T$ etc. Is it a matter of coincidence or is it always going to happen? Well there is something more deeper here in the sense that any two consecutive directions $d^{(k)}$ and $d^{(k+1)}$ given by the steepest descent method are mutually orthogonal (see Theorem 9.4.1 below). Therefore in $\mathbf{R}^2$, if the first two directions are $d^{(1)}$ and $d^{(2)}$ then $d^{(3)}$ has to be $d^{(1)}$ and $d^{(4)}$ has to be $d^{(2)}$ so on. But this repetition may not be in $\mathbf{R}^3$ and higher dimensional spaces because there if $d^{(1)}$ and $d^{(2)}$ are orthogonal then we may have $d^{(3)}$, different from $d^{(1)}$, which is orthogonal to $d^{(2)}$. So the important thing is the orthogonality of consecutive directions and NOT their alternate repetition.*

Theorem 9.4.1 *Let $d^{(k)}$ and $d^{(k+1)}$ be two consecutive directions generated by the steepest descent method. Then*

$$< d^{(k+1)}, d^{(k)} >= 0$$

where $<,>$ denotes the standard inner product in $\mathbf{R}^n$.

Proof. Let $\overline{\alpha_k} > 0$ such that $h(\overline{\alpha_k}) = \underset{\alpha_k>0}{Min}\, h(\alpha_k)$. Then

$$\left.\frac{d\,h}{d\,\alpha_k}\right|_{\alpha=\overline{\alpha_k}} = 0,$$

i.e. $$\left.\frac{d\,[f(x^{(k)} + \alpha_k\, d^{(k)})]}{d\,\alpha_k}\right|_{\alpha=\overline{\alpha_k}} = 0$$

i.e. $$\left.\left(\nabla\, f(x^{(k)} + \alpha_k\, d^{(k)})\right)^T\, d^{(k)}\right|_{\alpha=\overline{\alpha_k}} = 0$$

i.e. $$\left.\left(\nabla\, f(x^{(k)} + \alpha_k\, d^{(k)})\right)^T \left(-\frac{\nabla f(x^{(k)})}{\|\nabla f(x^{(k)})\|}\right)\right|_{\alpha=\overline{\alpha_k}} = 0$$

i.e. $$\left(\nabla\, f(x^{(k)} + \overline{\alpha}_k\, d^{(k)})^T\right) \nabla\, f(x^{(k)}) = 0$$

i.e. $$< d^{(k+1)}, d^{(k)} > = 0\ .$$

□

Remark 9.4.3 *A geometrical interpretation of the above theorem could be that the vector* $-\nabla\, f(x^{(k)})$ *which is normal to the surface* $f(x) =$ *constant at* $x^{(k)}$, *is tangent to the surface at the point* $x^{(k+1)}$. *Therefore in* $\mathbf{R}^2$, *the movement of the steepest descent method will be as shown in Fig 9.3*

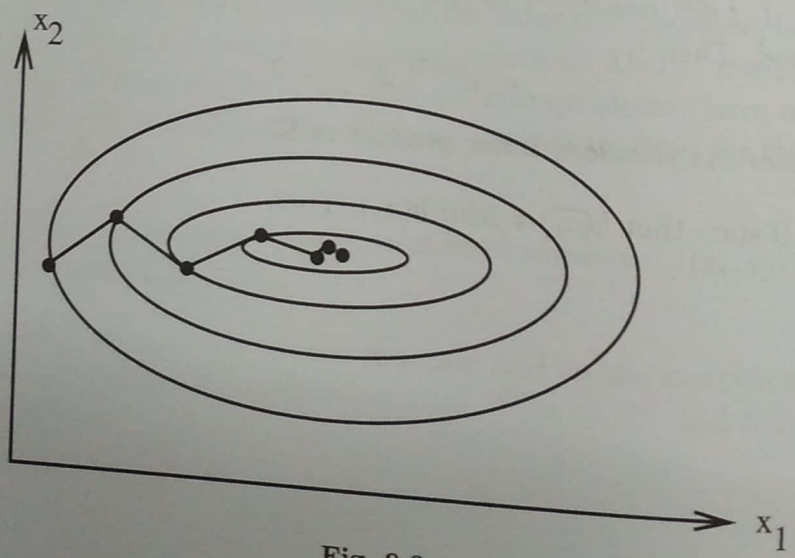


Fig. 9.3.

9.5 Newton's Method

Newton's method for finding real roots of the equation $g(y) = 0, y \in \mathbf{R}$ is well known. The basic scheme here is $y_{k+1} = y_k - g(y_k)/g'(y_k)$ where y_k is the current iterate or the current approximation.

It is natural here to be somewhat curious to know the connection between solving UMP's and the classical Newton's method for root finding. For this let us recollect that for solving the unconstrained minimization problem (9.3) we are aiming at finding a point $\bar{x} \in \mathbf{R}^n$ such that $\nabla f(\bar{x}) = 0$. Therefore the basic problem of root finding enters here very naturally except that rather than finding the roots of a single equation we have to find the roots of a system, namely $\nabla f(x) = 0$. Looking at the standard basic scheme of Newton's method for $g(y) = 0, y \in \mathbf{R}$, we immediately get the scheme

$$x^{(k+1)} = x^{(k)} - (H_f(x^{(k)}))^{-1} \nabla f(x^{(k)})$$

for finding a solution of the system $\nabla f(x) = 0$.

A more acceptable mathematical argument for the above scheme could be as follows. Let us approximate the given function f (note that $f : \mathbf{R}^n \longrightarrow \mathbf{R}$ is the given function in problem (9.3) which is to be optimized) in a neighborhood of the current approximate $x^{(k)}$ by the truncated Taylor series to get

$$f(x) \simeq f(x^{(k)}) + (x - x^{(k)})^T \nabla f(x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T H_f(x^{(k)})(x - x^{(k)}).$$

Therefore if we wish to minimize $f(x)$, it makes sense to minimize its quadratic approximation $q(x)$ where

$$q(x) = f(x^{(k)}) + (x - x^{(k)})^T \nabla f(x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T H_f(x^{(k)})(x - x^{(k)}).$$

Let this minimization be done exactly and hence

$$\nabla q(x) = 0,$$

i.e.

$$\nabla f(x^{(k)}) + \frac{1}{2}(x - x^{(k)})\, 2\, H_f(x^{(k)}) = 0$$

i.e.

$$(x - x^{(k)}) = -(H_f(x^{(k)}))^{-1} \nabla f(x^{(k)}),$$

which gives

$$x^{(k+1)} = x^{(k)} - (H_f(x^{(k)}))^{-1} \nabla f(x^{(k)}).$$

Since Newton's method for solving problem (9.3) essentially finds the roots of the system $\nabla f(x) = 0$, it follows that for minimizing a positive definite quadratic form of n variables, it will take exactly one iteration (imagine finding the roots of a linear

equation by Newton's method) because the system $\nabla f(x) = 0$ will be a linear system of equations. Therefore Newton's method for solving UMP's has quadratic termination property (in fact here for minimizing a positive definite quadratic form of n variables it will take exactly one iteration). Also it will have order of convergence $p = 2$ and it will have descent property, but it will not have the property of global convergence (because the standard Newton's method for root finding does not have this property). Therefore except for the case when we are minimizing a positive definite quadratic form (in the context of root finding it means solving a linear equation), we cannot start the method from an arbitrary point $x^{(0)} \in \mathbf{R}^n$.

Therefore if we are minimizing a positive definite quadratic form by Newton's method, then not only that we can start from any arbitrary point $x^{(0)} \in \mathbf{R}^n$, we also know that we will get the optimal solution in exactly one iteration, i.e. $x^{(1)}$ has to be the optimal solution $\bar{x}$. However if the function f in problem (9.3) is not a positive definite quadratic form then there are major problems with Newton's method. Apart from the fact that in this situation we cannot start from an arbitrary starting point $x^{(0)}$ ($x^{(0)}$ has to be 'close' to $\bar{x}$), there are serious issues with regard to the Hessian $H_f(x^{(k)})$. Why should $H_f(x^{(k)})$ be invertible at the point $x^{(k)}$ for every k ? It may be reasonable to assume that $H_f(\bar{x})$ is invertible in a neighborhood of $\bar{x}$ (because f has to behave like a 'parabola' or a positive definite quadratic form around a point which is a strict local min point) but assuming its invertibility for every $x^{(k)}$ does not make sense. However Newton's method has order of convergence $p = 2$ and this is very attractive because there will be significant improvement in the value of the objective function even when we are close to the actual minimizing point. (Recall that the order of convergence of the steepest descent method is $p = 1$ which makes the algorithm very slow near the actual optimal point).

In view of the above we should try to make certain modifications in Newton's method and get a method (say modified Newton's method) which has all the nice properties of Newton's method (descent property, quadratic termination property and order of convergence $p = 2$) and is also globally convergent (so that we can start from an arbitrary point $x^{(0)} \in \mathbf{R}^n$). Also it will be nice if the proposed method also takes care of the issues related with the existence of the inverse of the Hessian. We discuss modified Newton's method in the next section.

Example 9.5.1 *Use Newton's method to minimize* $f(x_1, x_2) = 8x_1^2 - 4x_1x_2 + 5x_2^2$, $(x_1, x_2) \in \mathbf{R}^2$.

Solution As the function f is a positive definite quadratic form in two variables we know for certain that we can start from any arbitrary point $x^{(0)} \in \mathbf{R}^2$ and use Newton's method to get $x^{(1)}$, then $x^{(1)}$ has to be the minimizing point.

To be specific, let $x^{(0)} = (5, 2)^T$. Then

$$\nabla f(x^{(0)}) = \begin{pmatrix} 16x_1 - 4x_2 \\ -4x_1 + 10x_2 \end{pmatrix}\Big|_{(x_1=5, x_2=2)}$$
$$= \begin{pmatrix} 72 & 0 \end{pmatrix},$$
$$H_f(x^{(0)}) = \begin{pmatrix} 16 & -4 \\ -4 & 10 \end{pmatrix},$$

and

$$(H_f(x^{(0)}))^{-1} = \frac{1}{144}\begin{pmatrix} 10 & 4 \\ 4 & 16 \end{pmatrix}.$$

Then

$$x^{(1)} = x^{(0)} - (H_f(x^{(0)}))^{-1}\nabla f(x^{(0)})$$
$$= \begin{pmatrix} 5 \\ 2 \end{pmatrix} - \frac{1}{444}\begin{pmatrix} 10 & 4 \\ 4 & 16 \end{pmatrix}\begin{pmatrix} 72 \\ 0 \end{pmatrix}$$
$$= \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

giving $\bar{x}_1 = 0, \bar{x}_2 = 0$ as the minimizing point.

9.6 Modified Newton's Method

While discussing Newton's method in the last section we noted certain limitations as $H_f(x^{(k)})$ may not be invertible at the points $x^{(k)}$. There is something more to it - namely, even if we could guarantee the existence of $(H_f(x^{(k)}))^{-1}$, it is not necessary that $-(H_f(x^{(k)}))^{-1}\nabla f(x^{(k)})$ will be the direction of descent unless we could also guarantee that $H_f(x^{(k)})$ is positive definite at $x^{(k)}$. For this it is enough to check that $-(M_k(x^{(k)}))^{-1}\nabla f(x^{(k)})$ is always a direction of descent for any positive definite matrix M_k. Another difficulty with Newton's method has been its lack of global convergence property. Keeping these things in mind, the following modification to Newton's method is suggested

$$x^{(k+1)} = x^{(k)} - \overline{\alpha_k}\, M_k\, (\nabla f(x^{(k)})) \tag{9.5}$$

where M_k is an appropriate positive definite matrix (obtained from $H_f(x^{(k)})$ as explained here) and $\overline{\alpha_k} > 0$ is the step size which is chosen as in the steepest descent method, i.e. $\overline{\alpha_k} > 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\, h(\alpha_k),$$

where

$$h(\alpha_k) = f(x^{(k)} + \alpha_k\, d^{(k)}),\ d^{(k)} = -M_k(\nabla f(x^{(k)})).$$

Now comes the main question. How should we choose M_k given the matrix $H_f(x^{(k)})$? Here we should remember that $H_f(x^{(k)})$ may not be invertible at $x^{(k)}$ and if at some point

$x^{(k)}$ it is invertible it is not necessary that its inverse is positive definite. But $H_f(x^{(k)})$ is certainly real symmetric and hence all its eigen values are real. What we shall do now is to add a suitable matrix of the form $\epsilon_k I$ ($\epsilon_k > 0$) and take $M_k = (F_k)^{-1}$ where $F_k = (\epsilon_k I + H_f(x^{(k)}))$. Here $\epsilon_k > 0$ is to be chosen so that all eigenvalues of F_k become strictly positive and therefore F_k and M_k become positive definite. For this, given the point $x^{(k)}$, we fix a constant $\delta > 0$ and calculate all eigenvalues of $H_f(x^{(k)})$. Let ϵ_k be the smallest non negative constant for which all eigenvalues of the matrix $\epsilon_k I + H_f(x^{(k)})$ are greater than or equal to δ. Therefore, once $\epsilon > 0$ has been chosen in this manner, we take $F_k = (\epsilon_k I + H_f(x^{(k)}))$ and $M_k = (\epsilon_k I + H_f(x^{(k)}))^{-1}$.

It can be shown that with the above modification the method described above (called modified Newton's method) has all the nice properties, namely it has the descent property, it has quadratic termination property, it has property of global convergence and its order of convergence p is 2. However it is still not practical because to get M_k we need to compute all eigenvalues of $H_f(x^{(k)})$.

Therefore we now have two basic gradient based methods, namely the steepest descent method and the modified Newton's method, for solving unconstrained minimization problems. Whereas the steepest descent method is simple to implement, it is not very good from the convergence point of view, as its order of convergence p is one. The other method, namely the modified Newton's method has all the nice properties, including the global convergence and order two convergence, it is not of much use because of the effort involved in evaluating F_k and hence M_k. So the best option seems to be looking for those algorithms for UMP's which are somewhere in the middle of the spectrum, i.e. these are simpler to implement (unlike the modified Newton's method) and have better order of convergence (unlike the steepest descent method). There are a whole class of such methods, namely *conjugate direction methods* and *quasi Newton methods*. We shall discuss some of these in the next two sections.

9.7 The Conjugate Gradient Method

Here we present certain basic principles of conjugate direction methods for solving UMP's and discuss the conjugate gradient method in detail. As mentioned in the previous section, these methods are better than the steepest descent method (in terms of order of convergence) and are simpler to implement than the modified Newton's method.

Definition 9.7.1 (Conjugate Directions). *Let Q ne an $(n \times n)$ positive definite matrix. Any two non-zero vectors (directions) $d^{(1)}$, $d^{(2)} \in \mathbf{R}^n$ are said to be conjugate vectors or conjugate directions with respect to Q, if $(d^{(1)})^T Q d^{(2)} = 0$.*

Here we note that for $Q = I$, conjugacy reduces to the usual concept of orthogonality. Therefore if $d^{(1)}$ and $d^{(2)}$ are conjugate with respect to Q, sometimes we also call them Q - orthogonal.

The above definition is extended to more than two vectors in a natural manner, i.e. a set $\{d^{(0)}, \ldots, d^{(k)}\}$ of $(k+1)$ vectors in $\mathbf{R}^n$ is said to be conjugate if every two of them are so, i.e. $(d^{(i)})^T\ Q\ d^{(j)} = 0\ (i \neq j)$. Now onwards we shall not write 'conjugate with respect to Q' but rather just write 'conjugate' in case there is no confusion.

Result 9.7.1 *Let $\{d^{(0)}, d^{(1)}, \ldots, d^{(k)}\}$ be a set of $(k+1)$ non-zero vectors which are conjugate with respect to a given positive definite matrix Q. Then the vectors $d^{(0)}, d^{(1)}, \ldots, d^{(k)}$ are linearly independent.*

Proof. To prove the above result, we have to show that $\alpha_0\ d^{(0)} + \alpha_1\ d^{(1)} + \ldots + \alpha_k\ d^{(k)} = 0$ implies that each $\alpha_i = 0$. For this consider the equation

$$\alpha_0\ d^{(0)} + \alpha_1\ d^{(1)} + \ldots + \alpha_k\ d^{(k)} = 0$$

and pre-multiply both sides by $(d^{(i)})^T\ Q$. Then by the definition of conjugacy, we obtain

$$\alpha_i\ ((d^{(i)})^T\ Q\ d^{(i)}) = 0,$$

which gives $\alpha_i = 0$ as the matrix Q is positive definite. □

Now to understand the basic principles of conjugate direction methods, we first consider the quadratic case, i.e. the problem

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \quad \frac{1}{2}\ x^T Q\ x - b^T x \tag{9.6}$$

where Q is an $(n \times n)$ symmetric positive definite matrix. As Q is positive definite the objective function of problem (9.6) is strictly convex and therefore problem has unique minimizing point $\bar{x} \in \mathbf{R}^n$.

Further the KKT conditions for problem (9.6) give $\nabla\left(\frac{1}{2}\ x^T Q x - b^T x\right) = 0$, i.e. $Qx = b$, which implies $\bar{x} = Q^{-1}b$ (note that as Q is positive definite, Q^{-1} exists).

The above discussion shows that finding the unique minimizing point $\bar{x}$ of problem (9.6) is equivalent to finding the unique solution of the system of equations $Qx = b$, namely $\bar{x} = Q^{-1}b$. Hence there is no theoretical difficulty as Q^{-1} can always be computed and so $\bar{x}$ can always be determined. The main purpose of introducing conjugate directions is to obtain this unique $\bar{x}$ without really finding Q^{-1} explicitly. This is something we always attempt in numerical optimization, i.e. we do not compute the inverses explicitly. We may recall here that the idea of pivoting used in the simplex algorithm was introduced so that we do not compute the inverse of the basis matrix explicitly.

In the following result, we demonstrate that if we can get hold of a set of n non-zero vectors which are Q - conjugate, then the desired $\bar{x}$ can be obtained easily and this will not need the computation of Q^{-1} explicitly.

Result 9.7.2 *Let $\{d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}\}$ be a set of n non-zero vectors in $\mathbf{R}^n$ which are conjugate with respect to Q. Then $\bar{x}$, which is the unique solution to the system $Qx = b$ or equivalently, the unique minimizing point of problem (9.6), is given by*

$$\bar{x} = \sum_{k=0}^{n-1} \left(\frac{(d^{(k)})^T b}{(d^{(k)})^T Q d^{(k)}} \right) d^{(k)}. \tag{9.7}$$

Proof. Using Result 9.7.1, we note that the vectors $d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$ are linearly independent. As these vectors are exactly n in number, they form a basis of $\mathbf{R}^n$. Therefore there exist scalars $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ such that

$$\bar{x} = \alpha_0\, d^{(0)} + \alpha_1\, d^{(1)} + \ldots + \alpha_{n-1}\, d^{(n-1)}.$$

If we now pre-multiply the above equation by $(d^{(k)})^T Q$ and use the definition of conjugacy, we get

$$\alpha_k = \frac{(d^{(k)})^T Q \bar{x}}{(d^k)^T Q d^{(k)}}.$$

But $\bar{x}$ is the unique solution of $Q\,x = b$ i.e. $Q\,\bar{x} = b$ and hence

$$\alpha_k = \frac{(d^{(k)})^T b}{(d^{(k)})^T Q d^{(k)}}$$

which on substitution in the equation (9.7) gives the result. □

The above result can also be visualized as the output of an iterative process (see Theorem 9.7.1), which becomes very handy in describing the conjugate gradient method.

Theorem 9.7.1 (Conjugate Direction Theorem). *Let $\{d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}\}$ be a set of n non-zero vectors in $\mathbf{R}^n$ which are conjugate with respect to Q. For any $x^{(0)} \in \mathbf{R}^n$, the sequence $\{x^{(k)}\}$ generated according to*

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \overline{\alpha_k}\, d^{(k)}, \\ \overline{\alpha_k} &= -\frac{(g^{(k)})^T\, d^{(k)}}{(d^{(k)})^T\, Q\, d^{(k)}}, \\ g^{(k)} &= Qx^{(k)} - b, \end{aligned}$$

converges to the unique solution $\bar{x}$ of the system $Q\,x = b$ exactly after n steps, i.e. $x^{(n)} = \bar{x}$.

Proof. Since $\{d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}\}$ are Q - conjugate, they (as before) form a basis of $\mathbf{R}^n$, and so there exist scalars $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$ such that

$$\bar{x} - x^{(0)} = \lambda_0\, d^{(0)} + \lambda_1\, d^{(1)} + \ldots + \lambda_{n-1}\, d^{(n-1)}. \tag{9.8}$$

Now pre-multiplying the above equation by $(d^{(k)})^T Q$, we get

$$\lambda_k = \frac{(d^{(k)})^T Q(\bar{x} - x^{(0)})}{(d^{(k)})^T Q d^{(k)}}. \tag{9.9}$$

Also by following the iterative scheme as given in the hypothesis of the theorem, we get

$$\begin{aligned} x^{(1)} - x^{(0)} &= \alpha_0 d^{(0)} \\ x^{(2)} - x^{(0)} &= x^{(1)} + \alpha_1 d^{(1)} - x^{(0)} = \alpha_0\, d^{(0)} + \alpha_1\, d^{(1)} \\ &\vdots \end{aligned}$$

$$x^{(k)} - x^{(0)} = \alpha_0\, d^{(0)} + \alpha_1\, d^{(1)} + \ldots + \alpha_{k-1}\, d^{(k-1)}. \tag{9.10}$$

Again, the pre-multiplication of both sides of (9.10) by $(d^{(k)})^T Q$ gives

$$(d^{(k)})^T Q(x^{(k)} - x^{(0)}) = 0. \tag{9.11}$$

Therefore

$$\lambda_k = \frac{(d^{(k)})^T Q(\bar{x} - x^{(k)} + x^{(k)} - x^{(0)})}{(d^{(k)})^T Q d^{(k)}} \tag{9.12}$$

i.e,

$$\begin{aligned} \lambda_k &= \frac{(d^{(k)})^T Q(\bar{x} - x^{(k)})}{(d^{(k)})^T Q d^{(k)}} \text{ (by (9.11))} \\ &= \frac{(d^{(k)})^T (Q\bar{x} - Qx^{(k)})}{(d^{(k)})^T Q d^{(k)}} \\ &= \frac{(d^{(k)})^T (b - Qx^{(k)})}{(d^{(k)})^T Q d^{(k)}} \\ &= \frac{-(d^{(k)})^T g^{(k)}}{(d^{(k)})^T Q d^{(k)}} \\ &= \frac{-(g^{(k)})^T d^{(k)}}{(d^{(k)})^T Q d^{(k)}} \\ &= \alpha_k. \end{aligned}$$

Now by the iterative scheme

$$x^{(n)} - x^{(0)} = \alpha_0\, d^{(0)} + \alpha_1\, d^{(1)} + \ldots + \alpha_{n-1}\, d^{(n-1)}\,. \tag{9.13}$$

Also as given by (9.8),

$$\bar{x} - x^{(0)} = \lambda_0\, d^{(0)} + \lambda_1\, d^{(1)} + \ldots + \lambda_{n-1}\, d^{(n-1)}\,. \tag{9.14}$$

But as shown above, $\lambda_k = \alpha_k$ for all $k = 0, 1, \ldots, n-1$, and hence from (9.13)-(9.14)

$$x^{(n)} - x^{(0)} = \bar{x} - x^{(0)}$$

i.e.

$$x^{(n)} = \bar{x}.$$

In view of Result 9.7.1 and Theorem 9.7.1 we conclude that to solve problem (9.6) we need to obtain n non-zero conjugate directions $d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$. But how to determine these directions is the main question now. The applicability of conjugate direction methods will essentially depend on how simple or difficult is the method of finding these conjugate directions. In the following we present one such conjugate direction method where the conjugate directions are determined by using the gradient of the function f. This method therefore, is appropriately called *conjugate gradient method.*

Conjugate Gradient Method for the Quadratic Case

Let us again consider the unconstrained optimization problem (9.6), i.e.

$$\underset{x \in \mathbf{R}^n}{Min} \quad \frac{1}{2}\, x^T Q x - b^T x$$

where $b \in \mathbf{R}^n$ and Q is an $(n \times n)$ positive definite matrix. Let us also recall that solving the above problem is equivalent to finding the unique solution $\bar{x}$ of the system $Qx = b$. We first describe the conjugate gradient method and then later justify various steps involved therein.

Step 1 Choose $x^{(0)} \in \mathbf{R}^n$ arbitrary. Define $d^{(0)} = -g^{(0)} = b - Qx^{(0)}$. Set $k = 0$.

Step 2 Use the scheme

$$\begin{aligned}
x^{(k+1)} &= x^{(k)} + \overline{\alpha_k} d^{(k)} \\
\overline{\alpha_k} &= \frac{-(g^{(k)})^T d^{(k)}}{(d^{(k)})^T Q d^{(k)}} \\
d^{(k+1)} &= -g^{(k+1)} + \beta_k d^{(k)} \\
\beta_k &= \frac{(g^{(k+1)})^T Q d^{(k)}}{(d^{(k)})^T Q d^{(k)}} \\
g^{(k)} &= Qx^{(k)} - b\,.
\end{aligned}$$

Step 3 Continue till we get $x^{(n)}$. Then $x^{(n)} = \bar{x}$ and hence stop.

In the above we note that the first step is the same as in the steepest descent method. But there after, each successive step moves in a direction that is a linear combination of the current gradient and the preceding direction vector. Here the formula for the computation of the scalar β_k has been derived so as to guarantee that $d^{(k+1)}$ is conjugate with respect to all previous directions, i.e. $(d^{(k+1)})^T\ Q\ d^{(i)} = 0$, for $i = 0, 1, \ldots, k$. The main point to note here is that we do not need all directions $d^{(0)}, d^{(1)}, \ldots, d^{(k)}$ at one go. But, rather we start from $d^{(0)}$ and generate subsequent conjugate directions as we proceed with the algorithm. As soon as we determine $d^{(n-1)}$, the point $x^{(n)}$ is known and that is precisely the point $\bar{x}$. The mathematical justification of conjugate gradient follows from the conjugate direction theorem, i.e. Theorem 9.7.1.

Example 9.7.1 *Use the conjugate gradient method to minimize* $f(x_1, x_2) = 3x_1^2 - 4x_1x_2 + 2x_2^2 + 4x_1 + 6,\ (x_1, x_2) \in \mathbf{R}^2$.

Solution To use the conjugate gradient method we need to express the given quadratic function in the form $f(x) = \frac{1}{2}\, x^T Q x - b^T x$. It is simple to check here that $Q = \begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}$, $b = \begin{pmatrix} -4 \\ 0 \end{pmatrix}$ and $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

Further we also check that Q is positive definite so the given problem is exactly in the form of the unconstrained optimization problem (9.6). Also the objective function is a positive definite quadratic form in two variables, and therefore by the conjugate direction theorem we know that starting with any arbitrary point $x^{(0)} \in \mathbf{R}^2$ and following the steps of conjugate gradient method, once we compute $x^{(2)}$ then $x^{(2)}$ has to be the unique minimizing point $\bar{x}$. For the sake of illustration let $x^{(0)} = (0, 0)^T$.

Step 1 For $x^{(0)} = (0, 0)^T$, $g^{(0)} = Qx^{(0)} - b = \begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} -4 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$.

Therefore, $d^{(0)} = -g^{(0)} = (-4, 0)^T$.

Step 2 Next $x^{(1)} = x^{(0)} + \overline{\alpha_0} d^{(0)}$ where

$$\overline{\alpha_0} = \frac{-(g^{(0)})^T d^{(0)}}{(d^{(0)})^T Q d^{(0)}} = \frac{-(4, 0)\begin{pmatrix} 4 \\ 0 \end{pmatrix}}{(-4, 0)\begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}\begin{pmatrix} -4 \\ 0 \end{pmatrix}} = 1/6$$

Therefore,

$$x^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 1/6 \begin{pmatrix} -4 \\ 0 \end{pmatrix} = \begin{pmatrix} -2/3 \\ 0 \end{pmatrix}$$

Step 3 Now,

$$g^{(1)} = Qx^{(1)} - b = \begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}\begin{pmatrix} -2/3 \\ 0 \end{pmatrix} - \begin{pmatrix} -4 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 8/3 \end{pmatrix},$$

$$\beta_0 = \frac{(g^{(1)})^T\, Qd^{(0)}}{(d^{(0)})^T Q g^{(0)}} = \frac{(0, 8/3)\begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}\begin{pmatrix} -4 \\ 0 \end{pmatrix}}{96} = 4/9,$$

$$\text{and } d^{(1)} = -g^{(1)} + \beta_0 d^{(0)} = \begin{pmatrix} 0 \\ -8/3 \end{pmatrix} + \frac{4}{9}\begin{pmatrix} -4 \\ 0 \end{pmatrix} = \begin{pmatrix} -16/9 \\ -8/3 \end{pmatrix}.$$

Step 4 Now we obtain $x^{(2)}$ as
$x^{(2)} = x^{(1)} + \overline{\alpha_1} d^{(1)}$. But

$$\overline{\alpha_1} = \frac{-(g^{(1)})^T d^{(1)}}{(d^{(1)})^T Q d^{(1)}} = 3/4,$$

Therefore,

$$x^{(2)} = \begin{pmatrix} -2/3 \\ 0 \end{pmatrix} + \frac{3}{4}\begin{pmatrix} -16/9 \\ -8/3 \end{pmatrix}$$
$$= \begin{pmatrix} -2 \\ -2 \end{pmatrix}$$
$$= \overline{x},$$

i.e. the minimizing point of $f(x_1, x_2)$ is $\overline{x}_1 = -2$, $\overline{x}_2 = -2$.

Remark 9.7.1 *Here we can verify that $\nabla f(\overline{x})$, i.e. $Q\overline{x} - b = 0$ as it should be. Also $(d^{(1)})^T Q\, d^{(0)} = 0$. Again this has to be true because the basic argument of conjugate gradient method is to generate directions $d^{(k)}$ such that $d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$ are conjugate with respect to Q.*

Result 9.7.3 *For the conjugate gradient method, following are true*

(i) $(g^{(k+1)})^T d^{(k)} = 0$

(ii) $\overline{\alpha_k} = \dfrac{(g^{(k)})^T g^{(k)}}{(d^{(k)})^T Q\, d^{(k)}}$

(iii) $\beta_k = \dfrac{(g^{(k+1)})^T g^{(k+1)}}{(g^{(k)})^T g^{(k)}}$

(iv) $(d^{(k)})^T Q\, d^{(i)} = 0$, $(i = 0, 1, \ldots, k-1)$.

Here $g^{(k)}$ is the gradient of the objective function at $x^{(k)}$ i.e. $g^{(k)} = Q\, x^{(k)} - b$.

Proof. We shall prove (i), (ii) and (iii) only. As far as (iv) is concerned, from the definition of β_k, it is simple to prove that $(d^{(k+1)})^T Q d^{(k)} = 0$ and then to prove that $(d^{(k+1)})^T Q d^{(i)} = 0$ for all $i = 0, 1, \ldots, k-1$, we have to use induction. For full proof of this, we shall refer to Luenberger [106].

To prove (i) we note that $\overline{\alpha_k} > 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\ h(\alpha_k) = \underset{\alpha_k > 0}{Min}\ f(x^{(k)} + \alpha_k d^{(k)}),$$

and therefore,

$$\left.\frac{dh(\alpha_k)}{d\alpha_k}\right|_{\alpha_k = \overline{\alpha_k}} = 0,$$

i.e. $(\nabla f(x^{(k)} + \overline{\alpha_k} d^{(k)}))^T d^{(k)} = 0$
i.e. $(\nabla f(x^{(k+1)}))^T d^{(k)} = 0,$
i.e. $(g^{(k+1)})^T d^{(k)} = 0.$

Now
$$-(g^{(k)})^T d^{(k)} = -(g^{(k)})^T(-g^{(k)} + \beta_{k-1} d^{(k-1)})$$

$$= (g^{(k)})^T g^{(k)} - \beta_{k-1}(g^{(k)})^T d^{(k-1)}$$

$$= (g^{(k)})^T g^{(k)} - 0 = (g^{(k)})^T g^{(k)},$$

which proves the result (ii).

Next we shall prove (iii). For this we first note that the gradient vectors $\{g^{(k)}\}$ are mutually orthogonal, i.e. $(g^{(k+1)})^T g^{(k)} = 0$. This is because

$$Qx^{(k+1)} - b = Q\,(x^{(k)} + \overline{\alpha_k} d^{(k)}) - b,$$

i.e.

$$Qx^{(k+1)} - b = Q\,x^{(k)} - b + \overline{\alpha_k}\,Q\,d^{(k)},$$

i.e.

$$g^{(k+1)} = g^{(k)} + \overline{\alpha_k}\,Q\,d^{(k)},$$

i.e.

$$(g^{(k+1)})^T = (g^{(k)})^T + \overline{\alpha_k}\,d^{(k)^T}\,Q.$$

But $d^{(k)} = -g^{(k)} + \beta_{k-1} d^{(k-1)}$ and therefore
$(g^{(k+1)})^T g^{(k)} = (g^{(k+1)})^T[-d^{(k)} + \beta_{k-1} d^{(k-1)}]$

$$= -(g^{(k+1)})^T d^{(k)} + \beta_{k-1}(g^{(k+1)})^T d^{(k-1)}$$
$$= 0 + \beta_{k-1}(g^{(k+1)})^T d^{(k-1)}$$
$$= \beta_{k-1}(g^{(k+1)})^T d^{(k-1)}$$
$$= \beta_{k-1}[(g^{(k)})^T + \overline{\alpha_k}(d^{(k)})^T Q] d^{(k-1)}$$
$$= \beta_{k-1}[(g^{(k)})^T d^{(k-1)} + \overline{\alpha_k}(d^{(k)})^T Q d^{(k-1)}]$$
$$= \beta_{k-1}[0 + 0] = 0.$$

Also $g^{(k+1)} - g^{(k)} = Q(x^{(k+1)} - x^{(k)}) = Q(\overline{\alpha_k} d^{(k)})$
and therefore

$$Qd^{(k)} = \frac{1}{\overline{\alpha_k}}(g^{(k+1)} - g^{(k)}),$$

which gives

$$\beta_k = \frac{(g^{(k+1)})^T Q d^{(k)}}{(d^{(k)})^T Q d^{(k)}} = \frac{(g^{(k+1)})^T (g^{(k+1)} - g^{(k)})}{\overline{\alpha_k}((d^{(k)})^T Q d^{(k)})}$$

$$= \frac{(g^{(k+1)})^T g^{(k+1)} - (g^{(k+1)})^T g^{(k)}}{(d^{(k)})^T Q d^{(k)}} \times \frac{(d^{(k)})^T Q d^{(k)}}{(g^{(k)})^T g^{(k)}}$$

$$= \frac{(g^{(k+1)})^T g^{(k+1)} - 0}{(g^{(k)})^T g^{(k)}} = \frac{(g^{(k+1)})^T g^{(k+1)}}{(g^{(k)})^T g^{(k)}}$$

as desired. □

Fletcher and Reeves' Method

Let us consider the general unconstrained minimization problem

$$\underset{x \in \mathbf{R}^n}{Min} \quad f(x) \tag{9.15}$$

and attempt to translate the steps of the conjugate gradient method for problem (9.6) to problem (9.15). The first thing we note is that $g^{(k)}$ has to be taken as $\nabla f(x^{(k)})$ and as $\overline{\alpha_k} > 0$ is the step size, it has to be computed in the usual manner. In fact the closed form expression for $\overline{\alpha_k}$ for the quadratic case comes from the exact minimization of $h(\alpha_k)$, as for the quadratic case $h(\alpha_k)$ comes out to be a quadratic expression in α_k. Also rather than the expression for β_k in terms of Q, we can use the expression given by Result 9.7.3 which involves $g^{(k)}$ and $g^{(k+1)}$ only. Therefore it makes sense to have the following steps for solving problem (9.15).

Step 1 Choose $x^{(0)} \in \mathbf{R}^n$ arbitrary and take $d^{(0)} = -g^{(0)} = -\nabla f(x^{(0)})$. Set $k = 0$.

Step 2 Obtain $\overline{\alpha_k} > 0$ such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\, h(\alpha_k)\,,$$

where

$$h(\alpha_k) = f(x^{(k)} + \alpha_k d^{(k)})\,.$$

Step 3 Define $x^{(k+1)} = x^{(k)} + \overline{\alpha_k} d^{(k)}$ and take $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$ where $g^{(k+1)} = \nabla f(x^{(k+1)})$ and β_k is given by

$$\beta_k = \frac{(g^{(k+1)})^T g^{(k+1)}}{(g^{(k)})^T g^{(k)}} .$$

Step 4 Continue till we get a point $x^{(p)}$ of the solution set, i.e. $\|\nabla f(x^{(p)})\| < \epsilon$ for some preassigned tolerance $\epsilon > 0$.

We can show that the above steps produce a method which has the descent property and also has the quadratic termination property, but is not globally convergent, i.e. we can not start the method from an arbitrary point $x^{(0)} \in \mathbf{R}^n$. To make this algorithm globally convergent we incorporate *Powell's correction* in the above procedure and what we get is called the *Fletcher and Reeves' method* for solving the unconstrained minimization problem (9.15). The so called, Powell's correction essentially tells that starting with an arbitrary point $x^{(0)}$ perform Steps 2-4 as described above till we get $x^{(n)}$. Then we go back to Step 1 above replacing $x^{(0)}$ by $x^{(n)}$ and continue. Thus after every $(n-1)$ iterations, the direction is again taken as the steepest descent direction at the current point. Therefore the stepwise description of the Fletcher and Reeves' method is as follows

Step 1 Choose $x^{(0)} \in \mathbf{R}^n$ arbitrary and take $d^{(0)} = -g^{(0)} = -\nabla f(x^{(0)})$.
Step 2 For $k = 0, 1, \ldots, (n-1)$,

(a) Set $x^{(k+1)} = x^{(k)} + \overline{\alpha_k} d^{(k)}$, where $\overline{\alpha_k} > 0$ is chosen such that

$$h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\, h(\alpha_k),\ h(\alpha_k) = f(x^{(k)} + \alpha_k d^{(k)}) .$$

(b) Compute $g^{(k+1)} = \nabla f(x^{(k+1)})$.
(c) Unless $k = (n-1)$, set

$$d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)},$$

$$\beta_k = \frac{(g^{(k+1)})^T g^{(k+1)}}{(g^{(k)})^T g^{(k)}}.$$

Step 3 Replace $x^{(0)}$ by $x^{(n)}$ and go back to Step 1.

9.8 Davidon-Fletcher-Powell (DFP) Method

As in Section 9.7, here again we first consider the quadratic case, i.e. problem (9.6)

$$\underset{x \in \mathbf{R}^n}{Min}\ \ \frac{1}{2} x^T Q x - b^T x$$

where Q is an $(n \times n)$ positive definite matrix. Here the basic aim is to approximate the inverse of the Hessian $H_f(\overline{x})$, i.e. Q^{-1} by an appropriate sequence of positive definite

matrices and thereby avoiding the computation of Q^{-1} explicitly. We shall now describe the DFP method for the above problem with the general notations $g^{(k)}$, $\overline{\alpha_k}$ etc and then specify modifications for solving the unconstrained minimization problem (9.15).

Step 1 Choose $x^{(0)} \in \mathbf{R}^n$ arbitrary. Also choose any positive definite matrix. S_0 (we may take $S_0 = I$). Set $k = 0$.

Step 2 Define $d^{(k)} = -S_k g^{(k)}$, where $g^{(k)} = \nabla f(x^{(k)})$. (For the quadratic case $\nabla f(x) = Qx - b$)

Step 3 Obtain $\overline{\alpha_k} > 0$ such that $h(\overline{\alpha_k}) = \underset{\alpha_k > 0}{Min}\, h(\alpha_k)$ where $h(\alpha_k) = f(x^{(k)} + \alpha_k d^{(k)})$. Define $x^{(k+1)} = x^{(k)} + \overline{\alpha_k} d^{(k)}$.

Step 4 Set $p^{(k)} = x^{(k+1)} - x^{(k)} = \overline{\alpha_k} d^{(k)}$ and $q^{(k)} = g^{(k+1)} - g^{(k)}$. Update S_k to S_{k+1} as

$$S_{k+1} = S_k + \frac{p^{(k)}(p^{(k)})^T}{(p^{(k)})^T q^{(k)}} - \frac{S_k\, q^{(k)}\, (q^{(k)})^T S_k}{(q^{(k)})^T\, S_k q^{(k)}}\,.$$

Step 5 Continue till we get $x^{(n)}$. In that case $x^{(n)} = \overline{x}$, the unique minimizing point of problem (9.6).

Remark 9.8.1 *We can prove that $\{S_k\}_0^n$ is a sequence of positive definite matrices and $S_n = Q^{-1}$. Also $(p^{(i)})^T Q p^{(j)} = 0$, for $0 \le i < j < k$ and therefore $p^{(k)}$'s are conjugate directions with respect to Q. Since from the current point $x^{(k)}$ we are moving in the direction $p^{(k)} = \overline{\alpha_k} d^{(k)}$, we observe that the DFP method can also be thought of as a conjugate direction method. Further if we take $S_0 = I$, then it really becomes the conjugate gradient method. Here it should be noted that it is not necessarily true that starting from the same $x^{(0)} \in \mathbf{R}^n$, with $S_0 = I$, the DFP method and the conjugate gradient method will generate the same conjugate directions.*

Remark 9.8.2 *It can be proved that matrices A_k and B_k are real symmetric matrices of rank 1, where*

$$A_k = \frac{p^{(k)}(p^{(k)})^T}{(p^{(k)})^T q^{(k)}},$$

and

$$B_k = \frac{S_k q^{(k)} (q^{(k)})^T S_k}{(q^{(k)})^T S_k q^{(k)}}\,.$$

Therefore the updation formula $S_{k+1} = S_k + A_k - B_k$ is called a rank one updation.

Remark 9.8.3 *We can also prove that $S_{k+1} Q\ p^{(i)} = p^{(i)}$, $0 \le i \le k$. Therefore $p^{(0)}, p^{(1)}, \ldots, p^{(k)}$ are eigen vectors corresponding to unity eigen value for the matrix $S_{k+1}Q$. These eigen vectors are linearly independent as they are conjugate with respect to Q. Therefore $S_n = Q^{-1}$.*

Remark 9.8.4 *Since (DFP) method is also a conjugate gradient type method (for $S_0 = I$), it will have the same properties, i.e. it has the descent property, it has the quadratic termination property but will not have the property of global convergence in general. However by incorporating the Powell's correction this method can also be made globally convergent for solving the unconstrained optimization problem (9.15). Thus similar to Fletcher and Reeves method for UMP's, after every n steps, once we get $x^{(n)}$, we restart the method replacing $x^{(0)}$ by $x^{(n)}$ and S_0 by S_n.*

Example 9.8.1 *Use the DFP method to minimize $f(x_1, x_2)$ over $(x_1, x_2) \in \mathbf{R}^2$ where, $f(x_1, x_2) = 3x_1^2 - 4x_1x_2 + 2x_2^2 + 4x_1 + 6$, starting with $x^{(0)} = (0,0)^T$ and $S_0 = I$.*

Solution The given function is a positive definite quadratic form and the given problem has the form of problem (9.6) with

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, c = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, Q = \begin{pmatrix} 6 & -4 \\ -4 & 4 \end{pmatrix}.$$

Step 1 Take $x^{(0)} = (0,0)^T$, $S_0 = I$ and therefore

$$d^{(0)} = -S_0 g^{(0)} = -\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 4 \\ 0 \end{pmatrix} = \begin{pmatrix} -4 \\ 0 \end{pmatrix}.$$

Step 2 Now we find $\overline{\alpha_0}$. For this we have

$h(\alpha) = f(x^{(0)} + \alpha_0 d^{(0)}) = f(-4\alpha_0, 0)$, we have

$f(-4\alpha_0, 0) = 48\alpha_0^2 + 16\alpha_0 + 6$. Therefore

$h(\overline{\alpha}) = \underset{\alpha_0 > 0}{Min}\, h(\alpha_0)$ gives $\overline{\alpha_0} = 1/6$.

Step 3 We next obtain

$$x^{(1)} = x^{(0)} + \overline{\alpha_0} d^{(0)} = \begin{pmatrix} -2/3 \\ 0 \end{pmatrix}$$

$$p^{(0)} = x^{(1)} - x^{(0)} = \begin{pmatrix} -2/3 \\ 0 \end{pmatrix}$$

$$q^{(0)} = g^{(1)} - g^{(0)} = \begin{pmatrix} 0 \\ 8/3 \end{pmatrix} - \begin{pmatrix} 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ -8/3 \end{pmatrix}$$

$$A_0 = = \begin{pmatrix} 1/6 & 0 \\ 0 & 0 \end{pmatrix}$$

$$B_0 = \begin{pmatrix} 9/13 & -6/13 \\ -6/13 & 4/13 \end{pmatrix}$$

$$S_1 = S_0 + A_0 - B_0 = \begin{pmatrix} 37/78 & 6/13 \\ 6/13 & 9/13 \end{pmatrix}.$$

Step 4 The next direction $d^{(1)}$ is given by $d^{(1)} = -S_1 g^{(1)} = -\begin{pmatrix} 16/13 \\ 24/13 \end{pmatrix}$. Further $\overline{\alpha_1} = 13/12$ is obtained by minimizing $h(\alpha_1) = f(x^{(1)} + \alpha_1 d^{(1)})$ over $\alpha_1 > 0$. Therefore

$$x^{(2)} = x^{(1)} + \overline{\alpha_1} d^{(1)} = \begin{pmatrix} -2 \\ -2 \end{pmatrix} = \overline{x}.$$

Here we can verify some of the results stated earlier. For example. we can check that the matrices A_0 and $-B_0$ are symmetric matrices of rank one and the matrix S_1 is positive definite. Further if we also evaluate $p^{(1)}$ and $q^{(1)}$ to get S_2, then $S_2 = Q^{-1}$ and $(p^{(1)})^T Q p^{(0)} = 0$.

9.9 Preconditioning

Let Q be an $(n \times n)$ positive definite matrix. Then we have seen that minimizing $\frac{1}{2}x^T Qx - b^T x$ over $x \in \mathbf{R}^n$, is equivalent to finding the unique solution of the system $Qx = b$. In this context, it is known that (e.g. Luenberger [106]) the convergence rate actually depends on the ratio $r = \frac{\lambda_{max}}{\lambda_{min}}$, where λ_{max} and λ_{min} are the largest and the smallest eigen values of Q. The number r is called the *condition number* of the matrix Q. The convergence is best when r is close to one and it is becomes slow as r increases. This is because for $r = 1$, i.e. $\lambda_{max} = \lambda_{min}$, the contours of the objective function becomes circular. Therefore to accelerate the convergence of an algorithm we should try to modify the eigenvalue structure by transforming the matrix Q suitably.

Let the given system be $Qx = b$, where Q is an $(n \times n)$ positive definite matrix, $x \in \mathbf{R}^n$ and $b \in \mathbf{R}^m$. The key idea of preconditioning is to change the variable x to v via a nonsingular matrix C, $v = Cx$. The quadratic form $(\frac{1}{2}x^T Qx - b^T x)$ then gets changed to $\frac{1}{2}(v^T((C^{-1})^T Q C^{-1})v) - (C^{-1}b)^T v$. If we now use an algorithm (e.g. conjugate gradient method) to minimize this transformed quadratic form or equivalently solve the system $((C^{-1})^T Q C^{-1})v = C^{-1}b$, then the convergence rate will depend on the eigenvalue structure of $(C^{-1})^T Q C^{-1}$ rather than that of Q. Therefore in preconditioning our aim is to choose the matrix C such that eigen values of $(C^{-1})^T Q C^{-1}$ are almost equal.

In practice, it is not necessary, to carry out the transformation $v = Cx$ explicitly, because we can apply the algorithm to the problem

$$\text{Min} \quad \frac{1}{2}(v^T((C^{-1})^T Q C^{-1})v) - (C^{-1}b)^T v$$

to get $\overline{v}$ and then invert the transformation to get $\overline{x} = C\overline{v}$. In the case of the conjugate gradient method, we do not need C explicitly but rather use the matrix $M = C^T C$, which is symmetric and positive definite by construction. Such a method is called the preconditioned conjugate gradient method. We may refer to Nocedal and Wright [119] for further details in this regard.

9.10 Summary and Additional Notes

- In this Chapter we have discussed several iterative procedures for solving smooth unconstrained minimization problems. These procedures involve finding a direction for movement and a line search method to find the step size.

- Two basic line search methods namely, the golden section rule and the Fibonacci search method are discussed in Section 9.3 and it is shown that for the same numbers of function evaluations, the Fibonacci search method is 'optimal' in the sense that it will give a search interval which is almost 17 % smaller than the golden section rule.
- The steepest descent method and Newton's method are discussed in Sections 9.4 and 9.5 respectively. These methods form the basis of other gradient based methods, in particular, the conjugate gradient method and the DFP method, discussed in Sections 9.7 and 9.8 respectively.
- The steepest descent method uses the first order approximation of the function and it does not perform well when we are close to the optimum. Newton's method uses second order approximation of the function and it performs well even when we are close to the optimum. However, the convergence for Newton's method is guaranteed only when the starting iterate is chosen close to the optimal solution.
- M. R. Hestenes and E. Stiefel in 1952 originally gave the idea of the conjugate direction method which led to the development of the conjugate gradient method for the quadratic case. Later in 1964, R. Fletcher and C. Reeves extended the conjugate gradient method for nonlinear functions.
- The idea of the DFP method is originally due to W. C. Davidon in 1959, which was simplified and reformulated by R. Fletcher and M. Powell in 1963. This method is also referred to as the *variable metric method* and falls under the class of *quasi Newton methods*.
- The updation formula of the DFP method has led to several new updations of the current positive definite matrix, in particular the *BFGS class* given by C. Broyden, R. Fletcher, D. Goldfarb and D. Shanno.
- Although we have discussed only the step length based methods, there are another class of methods, developed in 70's, called the *trust region methods*. Here the step length α_k is always taken as unity, so that the new iterate is given by $x^{(k+1)} = x^{(k)} + d^{(k)}$. In order to ensure that the descent property holds, we may have to take several trial vectors before finding the satisfactory $d^{(k)}$. This involves solving a constrained sub problem of the form '$\underset{\alpha}{Min}\ (g^k)^T d + \frac{1}{2} d^T H_k d$, subject to $\|d\|_2 \leq \Delta$ for some Δ'. We may refer to Dixon [49], Gill et al. [68], Bonnans et al [22], and Nocedal and Wright [119] for further details.

9.11 Exercises

9.1 *Consider the function $f(x)$ defined over [0,4] as follows*

$$f(x) = \begin{cases} 2 - x, & 0 \leq x \leq 1 \\ x^2, & 1 < x \leq 2 \\ x + 2, & 2 < x \leq 4 \end{cases}$$

1. *Sketch $f(x)$ and hence justify that f is a unimodal min function.*
2. *Perform 4 iterations of the golden section rule to minimize $f(x)$ over [0,4].*

9.2 *Consider the problem of minimizing $\phi(x)$ over [-5,5] where*

$$\phi(x) = max\left(2-x, x-1, 1-\frac{x}{2}\right),\ x \in \mathbf{R}.$$

1. *Solve the above problem graphically.*
2. *Is ϕ a unimodal min function ? Give reasons.*
3. *Perform 3 iterations of the golden section rule and hence give an approximate value of x_{min}.*

9.3 *Complete 3 more iterations of the following table where the function being minimized is $|x|$.*

k	$x_{L,k}$	$x_{U,k}$	$x_{p,k}$	$x_{q,k}$	$E_{p,k}$	$E_{q,k}$	L/R
1	-5	2.64	-2.08	-0.29			

9.4 *Let* $\phi(x) = max\left(x^2, \frac{(1-x)}{2}\right), x \in \mathbf{R}$.

1. *Verify that ϕ is a unimodal min function.*
2. *Perform 3 iterations of the golden section rule to minimize $\phi(x)$ over [-1,1]. Identify the interval I_4 after 3 iterations which will contain the minimizing point.*
3. *Perform 3 iterations of the Fibonacci search method to minimize $\phi(x)$ over [-1,1], and hence identify the interval I_4.*
4. *compare the ratio $\frac{I_4}{I_1}$ as obtained at (2) and (3) above.*

9.5 *The Fibonacci search method is to be used to find within 10% the value of x in the interval [0,1] that maximizes the function $f(x) = Min(x, 2-x^2), x \in \mathbf{R}$.*

1. *How many iterations are required?*
2. *Give results for the first 3 iterations only.*

9.6 *Consider the problem of minimization of $f(x_1, x_2) = 4x_1^2 + 6x_2^2 - 8x_1x_2$ over $(x_1, x_2) \in \mathbf{R}^2$.*

1. *Perform 3 iterations of the steepest descent method taking the starting point with $x^{(0)} = (1,1)^T$.*
2. *Solve the given problem by Newton's method starting with $x^{(0)} = (-5, 10)$. How many iterations are needed to get the minimizing point? Give reasons for your answer.*
3. *Solve the given problem by the conjugate gradient method starting with $x^{(0)} = (-1,2)^T$. How many iterations are needed to get the minimizing point? Give reasons for your answer.*

4. *Solve the given problem by the DFP method starting with* $x^{(0)} = (-1,2)^T$. *Do you take the same number of iterations as in the conjugate gradient method? Give reasons for your answer.*

9.7 *Let* $d^{(1)} = (1,0)^T$ *and* $Q = \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix}$. *Find* $d^{(2)} \in \mathbf{R}^2$ $(d^{(2)} \neq 0)$ *such that* $d^{(1)}$ *and* $d^{(2)}$ *are conjugate directions with respect to* Q. *Hence find the minimum value of* $(x_1^2 + 3x_2^2 + 2x_1x_2 - x_1 - x_2)$.

9.8 *Consider the system*

$$x_1 - x_2 = -4$$

$$3x_1 - x_2 = 0\,.$$

Taking $d^{(1)} = (1,0)^T$, *use the conjugate direction method to find its solution.*

9.9 *Let* $v^{(1)}$, $v^{(2)}$, *and* $v^{(3)}$ *be linearly independent vectors in* $\mathbf{R}^3$ *and* Q *be a* (3×3) *positive definite matrix. Let*

$$d^{(1)} = v^{(1)}$$

$$d^{(2)} = v^{(2)} - \left(\frac{(d^{(1)})^T\ Qv^{(2)}}{(d^{(1)})^T\ Qd^{(1)}}\right) d^{(1)}$$

$$d^{(3)} = v^{(3)} - \left(\frac{(d^{(1)})^T\ Qv^{(3)}}{(d^{(1)})^T Qd^{(1)}}\right) d^{(1)} - \left(\frac{(d^{(2)})^T Qv^{(3)}}{(d^{(2)})^T Q\ d^{(2)}}\right) d^{(2)}\,.$$

1. *Show that* $d^{(1)}, d^{(2)}, d^{(3)}$ *constitutes a set of* Q*-conjugate directions in* $\mathbf{R}^3$.
2. *Verify your answer for (1) above for* $v^{(1)} = (1,0,0)^T$, $v^{(2)} = (1,1,0)^T$, $v^{(3)} = (1,1,1)^T$ *and* $Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$.
   Hence or otherwise use conjugate directions method to solve the system $Q\,x = b$, *where* $b = (0,1,-1)^T$.
3. *Generalize the above result for* $\mathbf{R}^n$.

9.10 *Use the conjugate gradient method to find a solution of the system*

$$x_1 + x_2 = 8$$

$$2x_1 + x_2 = 10\,.$$

You may take the starting point as $x^{(0)} = (0,0)^T$.

9.11 *Solve 9.10 above by the DFP method.*

9.12 *Use the conjugate direction method in the closed form to find the solution of*

$$-x_1 + 4x_2 = 2$$

$$4x_1 - x_2 = 7 .$$

You may taken one direction as $d^{(0)} = (1,0)^T$.
Write the equivalent UMP for the given system and solve the same by Newton's method.

9.13 *Show that the eigenvectors corresponding to distinct eigenvalues of a real symmetric matrix* Q *are conjugate with respect to* Q.

9.14 *Let* $Q = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ *and* $d^{(0)} = (1,0)^T$. *Find* $d^{(1)} \neq 0$ *such that* $d^{(0)}$ *and* $d^{(1)}$ *are conjugate with respect to* Q. *Is* $d^{(1)}$ *unique?*
Hence or otherwise use the conjugate direction method to solve the system

$$x_1 + 2x_2 = 4$$

$$3x_1 + x_2 = 7 .$$

9.15 *Are the following statements true? Give reasons for your answer.*

1. *Let* $\phi(x) = Min(3x - 10, -5x + 5)$ *for* $0 \leq x \leq 5$. *Then* ϕ *is a unimodal max function.*
2. *Let* $I_1 = 1$ *and* $I_n = 0.01$. *Then for the Fibonacci search method* $x_{p,1} = 0.382$, *and* $x_{q,1} = 0.618$.
3. *Let* $f(x_1, x_2) = 2(x_1x_2 - x_1^2 - 2x_2^2)$ *be maximized over* $\mathbf{R}^2$ *by Newton's method. Then, irrespective of the starting point, the method will always give* x_{max} *in exactly one iteration.*
4. *Let* $I_1 = 2$ *and* $I = 0.01$. *Then the value of* n *to be chosen for the Fibonacci search method is 10.*
5. *Let a function of three variables be minimized by the steepest descent method. Then* $d_k = \left(0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)^T$ *and* $d_{k+1} = \left(\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}\right)^T$ *can not be two consecutive directions of descent.*
6. *Let* $f : [-5, 20] \to \mathbf{R}$ *be a unimodal min function with* $f(4) = 10$ *and* $f(6) = 20$. *Then the point* x_{min} *lies in the interval [-5,4].*
7. *Let the function* $f(x_1, x_2) = (x_1^2 + 2x_2)$ *be minimized by the steepest descent method. Then the direction of descent at the point (0,1) is (0,-1).*
8. *The directions* $d^{(1)} = (1,2)^T$, $d_2 = (3,4)^T$ *and* $d_3 = (-1,1)^T$ *cannot be conjugate with respect to any positive definite matrix* Q.
9. *Let* Q *be an* $(n \times n)$ *positive definite matrix then there cannot be more than* n *conjugate directions with respect to* Q.
10. *For minimizing the function* $2x_1^2 + 3x_2^2 + 2x_3^2 + 2x_1x_2$, *starting with* $(x_1^0 = -2, x_2^0 = -3, x_3^0 = 1)$, *Newton's method will give the minimizing point exactly after two iterations.*

9.16 *Fill in the blanks.*

1. *For minimizing the function* $(x_1^2 + 2x_2)$ *by the steepest descent method, the direction of descent at the point* $(0,1)$ *is* .....
2. *For minimizing the function* $6x_1^2+x_2^2+4x_1x_2$, *the DFP method will give a point* $(\overline{x_1}, \overline{x_2})$ *where the gradient is* .....
3. *If* $f(x) = x^3 + ax^2 + bx$ *has a local maximum at* $x = -1$ *and a local minimum at* $x = 1$, *then* a=..... *and* b=.....
4. *If* $I_1 = 10$ *and* $I_n = 0.001$, *then for using the Fibonacci search method, the value of* n *should be* .....
5. *The directional derivative of* $f(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2$ *at* $(1,0)$ *in the direction of* $\nabla f\big|_{(1,0)}$ *is* .....
6. *For a function* $f : \mathbf{R}^n \to \mathbf{R}$, *let* $(\nabla f(\overline{x}))^T d \geq 0$ *for all directions* $d \in \mathbf{R}^n$, *then* $\nabla f(\overline{x})$ = ...
7. *The problem of minimizing* $f(x_1, x_2) = 6x_1 + 3x_1^2 + x_2^2 + 2x_1x_2$ *over* $\mathbf{R}^2$ *is equivalent to solving a system, of linear equations* $Qx = b$, *where* Q =....., b =.....
8. *Let the eigenvalue of a* 3×3 *real symmetric matrix* A *be 0.4, 1.2, and 0.4. Let* $B = A^2 - 2A + I$. *Then the condition number of* B *is* .....
9. *Let* $\{r_k\}$, $r_k = a_k$, $0 < a < 1$, *be the given sequence. Then* $\{r_k\}$ *converges to zero with the order of convergence as* .....
10. *The function* $f(x) = min(1-|x|, \sqrt{1-(x-1)^2})$, $0 \leq x \leq 1$, *is a unimodal* .....*functic but not a unimodal* .....*function.*

10 Algorithms in Nonlinear Programming

10.1 Introduction

In Chapter 7, we have studied Wolfe's method for solving a very special nonlinear programming problem, namely the quadratic programming problem (QPP). In this chapter we wish to proceed further and aim to develop algorithms for solving general nonlinear programming problems. For this, we first consider the class of linearly constrained NLP's and then study the class of those NLP's which may also be nonlinearly constrained.

There are several algorithms in the literature for solving NLP's, each having its own merits and demerits. Since an exhaustive discussion of these algorithms in a text book format is neither possible nor even desirable, we have to make our own judgement on the choice of various algorithms to be discussed in this chapter.

For solving linearly constrained NLP's we have selected only two methods for discussion in this chapter. These are commonly known as *Frank and Wolfe's Method* and *Rosen's gradient projection method.* The main reason for choosing Frank and Wolfe's method is that it is a simplex based method and has guaranteed convergence. Here the problem is solved by solving a sequence of linear programming problems which are constructed by obtaining the linear approximation of the objective function $f(x)$ at the current feasible point. On the other hand, Rosen's gradient projection method has a natural appeal because it is essentially a modification of the gradient based method which we have already studied for solving the unconstrained optimization problems.

A very popular method for solving general NLP's is the *sequential unconstrained minimization technique (SUMT)* due to Fiacco ad Mc-Cormick [57]. As the name suggests, here we solve the given nonlinear programming problem by solving a sequence of unconstrained minimization problems which we have already studied. Traditionally, there are two approaches for applying SUMT, one is called the *penalty function method*, and the other is called the *barrier function method*, and we plan to discuss both of these in the present chapter.

Certain optimization problems, e.g. the separable nonlinear programming problems, can be visualized as multistage decision problems. These problems can be solved efficiently by using the technique of *dynamic programming* due to R.E. Bellman. We present a very brief discussion on dynamic programming in this chapter.

Although we have not yet explicitly mentioned but, in view of our discussions in earlier chapters, it must be clear that for developing a satisfactory solution procedure for solving NLP's we must make appropriate smoothness and convexity assumptions on the objective and constraints functions.

10.2 Frank and Wolfe's Method

Let the given linearly constrained nonlinear programming problem be

$$\begin{aligned} \text{Min} \quad & f(x) \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0, \end{aligned} \tag{10.1}$$

where $x \in \mathbf{R}^n, b \in \mathbf{R}^m, A \in \mathbf{R}^{m \times n}$ and f is a differentiable convex function.

In (10.1), the linearity of the constraints guarantees that its feasible region is a convex set and convexity of the objective function f gives that every local min point is a global min point. Let the staring solution $x^{(0)}$ be taken as any feasible point of problem (10.1). Since the constraints of (10.1) are linear, the starting point $x^{(0)}$ can always be obtained by employing the Phase-I of the simplex algorithm to the set of constraints $Ax = b,\ x \geq 0$. Let $x^{(k)}$ be the current solution and $f_L(x)$ be the linear approximation of $f(x)$ in an appropriate neighbourhood of $x^{(k)}$. Then by Taylor's formula

$$f_L(x) = f(x^{(k)}) + (x - x^{(k)})^T \nabla f(x^{(k)}),$$

i.e.

$$f(x) \simeq f_L(x) = x^T \nabla f(x^{(k)}) + f(x^{(k)}) - (x^{(k)})^T \nabla f(x^{(k)})$$

in a suitable neighbourhood of $x^{(k)}$.

Now as we wish to minimize $f(x)$ it makes sense to minimize its linear approximation $f_L(x)$. But in the expression of $f_L(x)$ the terms $f(x^{(k)})$ and $(x^{(k)})^T \nabla f(x^{(k)})$ are constants for the given point $x^{(k)}$. Therefore to minimize $f_L(x)$ we should minimize $x^T \nabla f(x^{(k)})$ only. This motivates us to construct the following linear programming problem.

$$\begin{aligned} \text{Min} \quad & w_k(x) = x^T \nabla f(x^{(k)}) \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0\,. \end{aligned} \tag{10.2}$$

Let $\overline{x}^{(k)}$ be an optimal solution of (10.2). Then $w_k(\overline{x}^{(k)}) \leq w_k(x^{(k)})$. Thus there are two cases to be considered,

(i) $w_k(\overline{x}^{(k)}) = w_k(x^{(k)})$;
(ii) $w_k(\overline{x}^{(k)}) < w_k(x^{(k)})$.

In case (i) we shall be proving that $x^{(k)}$ satisfies the KKT conditions for problem (10.1) whereas in case (ii) we shall show the existence of a direction $d^{(k)}$ so that if we move in the direction $d^{(k)}$ from the current point $x^{(k)}$ then the objective function improves. Here it must be noted that, because of the convexity of f, case (i) will result in giving $x^{(k)}$ as an optimal solution of the given problem (10.1).

Let us consider the case (ii) first. We note that $w_k(\overline{x}^{(k)}) < w_k(x^{(k)})$ does not necessarily mean that $f(\overline{x}^{(k)}) < f(x^{(k)})$, because $f(x) \simeq f_L(x)$ holds in a neighborhood of $x^{(k)}$, and $\overline{x}^{(k)}$ may not lie in this neighborhood. However, if we define

$$x^{(k+1)} = (1-\alpha)x^{(k)} + \alpha\overline{x}^{(k)} = x^{(k)} + \alpha(\overline{x}^{(k)} - x^{(k)}), \quad 0 < \alpha \leq 1$$

then $x^{(k+1)}$ is feasible to problem (10.1) because $x^{(k)}$ and $\overline{x}^{(k)}$ are feasible and $x^{(k+1)}$ is a convex combination of $x^{(k)}$ and $\overline{x}^{(k)}$. Also, if we write $d^{(k)} = \overline{x}^{(k)} - x^{(k)}$ then $w_k(\overline{x}^{(k)}) < w_k(x^{(k)})$ means

$$(d^{(k)})^T \nabla f(x^{(k)}) < 0.$$

From the above, we infer that the directional derivative of f at $x^{(k)}$ in the direction $d^{(k)}$ is less than zero. Therefore $d^{(k)}$ is a good direction to move for seeking an improved value of f. Thus from $x^{(k)}$ we should move in the direction of $(\overline{x}^{(k)} - x^{(k)})$. To find that how much to move, i.e. the value of the *step size* $\overline{\alpha}_k$, we compute the minimum of $h(\alpha_k)$ over $(0, 1]$, i.e.

$$h(\overline{\alpha}_k) = \underset{\alpha_k \in (0,1]}{\text{Min}} \; h(\alpha_k)$$

where $h(\alpha_k) = f(x^{(k)} + \alpha_k d_k)$, $d^{(k)} = \overline{x}^{(k)} - x^{(k)}$. Thus the new point $x^{(k+1)}$ is given by

$$x^{(k+1)} = x^{(k)} + \overline{\alpha}_k \, d^{(k)} = x^{(k)} + \overline{\alpha}_k(\overline{x}^{(k)} - x^{(k)}) .$$

In the computation of $\overline{\alpha}_k$, we should note the main difference with respect to what we have been doing earlier while solving the unconstrained optimization problems. Here $h(\alpha_k)$ is being minimized over [0, 1] but for the unconstrained optimization problems it was only $\alpha_k > 0$ because the problem was unconstrained. Since (10.1) is a constrained problem there has to be some upper bound on α_k so that the new point $x^{(k+1)}$ does not go outside the feasible region. In the above we do not find the largest such α_k but rather by taking $0 < \alpha_k \leq 1$, we make sure that the point $x^{(k+1)}$ remains in the feasible region. Thus $x^{(k+1)}$ is an improved point but not necessarily the best improved point at this stage.

Stepwise Description of Frank and Wolfe's Method

The stepwise description of the Frank and Wolfe's method is now given below.

Step 1 Choose an initial starting point $x^{(0)}$ as any feasible point of the set of constraints $Ax = b,\ x \geq 0$. Set $k = 0$.

Step 2 Evaluate $\nabla f(x^{(k)})$ and construct the following linear programming problem

$$\begin{aligned} \text{Min} \quad & w_k(x) = x^T \nabla f(x^{(k)}) \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0. \end{aligned} \tag{10.3}$$

Let an optimal solution of (10.3) be $\bar{x}^{(k)}$.

Step 3 Evaluate $w_k(\bar{x}^{(k)})$ and $w_k(x^{(k)})$. If $w_k(\bar{x}^{(k)}) = w_k(x^{(k)})$, stop and get $x^{(k)}$ as an optimal solution of the given problem (10.1). Otherwise, i.e. if $w_k(\bar{x}^{(k)}) < w_k(x^{(k)})$, then go to Step 4.

Step 4 Obtain $\bar{\alpha}_k \in (0, 1]$ such that

$$h(\bar{\alpha}_k) = \min_{\alpha_k \in (0,1]} h(\alpha_k) = \min_{\alpha_k \in (0,1]} [f(x^{(k)} + \alpha_k(\bar{x}^{(k)} - x^{(k)}))].$$

Define

$$x^{(k+1)} = x^{(k)} + \bar{\alpha}_k\,(\bar{x}^{(k)} - x^{(k)})$$

and go to Step 2.

We now illustrate Frank and Wolfe's method with the help of Example 10.2.1 given below.

Example 10.2.1 *Use Frank and Wolfe's method to find an optimal solution of* $2x_1^2 + 2x_1x_2 + 2x_2^2 - 4x_1 - 6x_2$

subject to

$$\begin{aligned} x_1 + 2x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned} \tag{10.4}$$

Solution We note that the objective function is a convex function (infact it is a strict convex function) and the constraints are linear so the given problem can be solved by Frank and Wolfe's method. To start the method we need a feasible point $x^{(0)}$. In general such a point will be obtained by using the Phase-I of the simplex algorithm but here, as the problem is only in two variables, we can plot the feasible region and take any feasible point as a starting point. In particular we can take $x^{(0)} = (1/2, 1/2)^T$, as it is a feasible point.

Also we evaluate $\nabla f(x)$ at any general point x and get

$$\nabla f(x) = \begin{pmatrix} 4x_1 + 2x_2 - 4 \\ 2x_1 + 4x_2 - 6 \end{pmatrix}.$$

First Iteration

We have $\nabla f(x^{(0)}) = (-1,-3)^T$ and $w_0(x) = -x_1 - 3x_2$. Therefore we need to solve the following LPP

$$\begin{aligned} &\text{Min } w_0(x) = -x_1 - 3x_2 \\ &\text{subject to} \\ &\qquad x_1 + 2x_2 \le 2 \\ &\qquad x_1, x_2 \ge 0. \end{aligned} \tag{10.5}$$

Solving the above LPP graphically, we get $\bar{x}^{(1)} = (0,1)^T$, $w_0(\bar{x}^{(1)}) = -3$. Also $w_0(x^{(0)}) = -2$. Here, it may be remarked that we have solved the above LPP graphically because it involves only two variables. However, in general, we need to employ the simplex method to solve various LPP's occurring in Frank and Wolfe's method.

As $w_0(\bar{x}^{(1)}) < w_0(x^{(0)})$, $x^{(0)}$ is not a KKT point. Hence from $x^{(0)}$ we move in the direction $d_0 = (\bar{x}^{(1)} - x^{(0)})$ to get the new point $x^{(1)}$. For this we first obtain the function

$$h(\alpha_0) = f(x^{(0)} + \alpha_0(\bar{x}^{(1)} - x^{(0)})) = f\left(\frac{1-\alpha_0}{2}, \frac{1+\alpha_0}{2}\right)$$

$$= -4\left(\frac{1-\alpha_0}{2}\right) - 6\left(\frac{1+\alpha_0}{2}\right) + 2\left(\frac{1-\alpha_0}{2}\right)^2 + 2\left(\frac{1-\alpha_0}{2}\right)\left(\frac{1+\alpha_0}{2}\right) + 2\left(\frac{1+\alpha_0}{2}\right)^2,$$

and then choose $\bar{\alpha}_0 \in (0,1]$ such that $h(\bar{\alpha}_0) = \text{Min } h(\alpha_0)$ for $0 < \alpha_0 \le 1$. Here again, in general we may have to use a suitable one dimensional optimization algorithm, e.g the Fibonacci search method or the golden section rule. But for our example we may simply use ordinary calculations to get $\bar{\alpha}_0 = 1$. Therefore $x^{(1)} = \left(\frac{1-\bar{\alpha}_0}{2}, \frac{1+\bar{\alpha}_0}{2}\right)^T = (0,1)^T$.

Second Iteration

For the point $x^{(1)}$, $\nabla f(x^{(1)}) = (-2,-2)^T$ and hence we have construct the LPP

$$\begin{aligned} &\text{Min } w_1(x) = -2x_1 - 2x_2 \\ &\text{subject to} \\ &\qquad x_1 + 2x_2 \le 2 \\ &\qquad x_1, x_2 \ge 0. \end{aligned}$$

Now solving the above LPP we obtain $\bar{x}^{(1)} = (2,0)^T$ with $w_1(\bar{x}^{(1)}) = -4$ and $w_1(x^{(1)}) = -2$. As $w_1(\bar{x}^{(1)}) < w_1(x^{(1)})$, $x^{(1)}$ also is not a KKT point.

Therefore from $x^{(1)}$ we move in the direction $d^{(1)} = (\bar{x}^{(1)} - x^{(1)})$. For this we find $\bar{\alpha}_1$ by minimizing $h(\alpha_1)$ over $(0,1]$. But

$$h(\alpha_1) = f(x^{(1)} + \bar{\alpha}_1 (\bar{x}^{(1)} - x^{(1)})) = f(2\alpha_1, 1 - \alpha_1),$$

which gives $\bar{\alpha}_1 = 1/6$. Hence

$$x^{(2)} = x^{(1)} + \bar{\alpha}_1 (\bar{x}^{(1)} - x^{(1)}) = (1/3, 5/6)^T.$$

Third Iteration

For the point $x^{(2)}$ we obtain $\nabla f(x^{(2)}) = (-1, -2)^T$ and hence we construct the LPP

$$\begin{aligned} \text{Min} \quad & w_2(x) = -x_1 - 2x_2 \\ \text{subject to} \quad & \\ & x_1 + 2x_2 \leq 2 \\ & x_1, x_2 \geq 0. \end{aligned}$$

An optimal solution of the above LPP can be taken as $\bar{x}^{(2)} = (0, 1)^T$ (Note that for this LPP any point on the line segment joining points $(2, 0)^T$ and $(0, 1)^T$ is optimal). Now $w_0(\bar{x}^{(2)}) = -2$ and $w_0(x^{(2)}) = -2$. As $w_0(\bar{x}^{(2)}) = w_0(x^{(2)})$, $x^{(2)}$ is KKT point which, because of the convexity of f, becomes optimal for the given problem.

Therefore $\bar{x}_1 = 1/3$ and $\bar{x}_2 = 5/6$ is an optimal solution to the given problem (10.4).

10.3 Mathematical Justification of Frank and Wolfe's Method

Frank and Wolfe's method discussed in Section 10.2 can be justified mathematically provided we prove two things. Firstly we should prove that whenever $w_k(\bar{x}^{(k)}) = w_k(x^{(k)})$, $x^{(k)}$ is a KKT point, and secondly we should prove that the method always converges. We shall prove the first result fully here but the second result we shall only state and refer to an appropriate text e.g Zangwill [173].

Theorem 10.3.1 *Consider the linear programming problem (10.3). Let $\bar{x}^{(k)}$ be an optimal solution of (10.3) and $w_k(x^{(k)}) = w_k(\bar{x}^{(k)})$. Then $x^{(k)}$ is a KKT point of the given nonlinear programming problem (10.1).*

Proof. Problem (10.3) is given by

$$\begin{aligned} \text{Min} \quad & x^T \nabla f(x^{(k)}) \\ \text{subject to} \quad & \\ & Ax = b \\ & x \geq 0\,. \end{aligned}$$

As $\bar{x}^{(k)}$ is optimal to (10.3) and the constraints are linear, there exist KKT multipliers $\bar{\lambda} \in \mathbf{R}^m, \bar{\mu} \in \mathbf{R}^n_+$ such that the following KKT conditions hold

(i) $\nabla f(x^{(k)}) + A^T\overline{\lambda} - \overline{\mu} = 0$

(ii) $\overline{\mu}^T\overline{x}^{(k)} = 0$

(iii) $A\overline{x}^{(k)} = b$

(iv) $\overline{x}^{(k)} \geq 0, \overline{\mu} \geq 0.$

Now pre-multiplying the first equation by $(\overline{x}^{(k)})^T$ and using (iii) and (iv) we get following equivalent KKT conditions

(i) $\nabla f(x^{(k)}) + A^T\overline{\lambda} \geq 0$

(ii) $(\overline{x}^{(k)})^T\ (\nabla f(x^{(k)}) + A^T\overline{\lambda}) = 0$

(iii) $A\overline{x}^{(k)} = b$

(iv) $\overline{x}^{(k)} \geq 0.$

But $x^{(k)}$ is feasible, i.e. $Ax^{(k)} = A\overline{x}^{(k)} = b, x^{(k)} \geq 0$ also hold. Further, it is given that $w_k(x^{(k)}) = w_k(\overline{x}^{(k)})$ i.e. $(\overline{x}^{(k)})^T\nabla f(x^{(k)}) = (x^{(k)})^T\nabla f(x^{(k)})$. Hence the above set of KKT conditions become

(i) $\nabla f(x^{(k)}) + A^T\overline{\lambda} \geq 0$

(ii) $(x^{(k)})^T\ (\nabla f(x^{(k)}) + A^T\overline{\lambda}) = 0$

(iii) $Ax^{(k)} = b$

(iv) $x^{(k)} \geq 0,$

which are precisely the KKT conditions for (10.1) at the point $x^{(k)}$. This proves Theorem 10.3.1 □

Theorem 10.3.2 (Convergence Theorem). *Let f be continuously differentiable and the feasible region of problem (10.1) be nonempty, closed and bounded. Also, let $\{x^{(k)}\}$ be the sequence of points generated by Frank and Wolfe's method. Then $\{x^{(k)}\}$ is either finite and the last point of the sequence being a KKT point or each limit point of $\{x^{(k)}\}$ is a KKT point of problem (10.1).*

10.4 Rosen's Gradient Projection Method

This method is again motivated by the feasible direction philosophy but unlike Frank and Wolfe's method, it does not require the solution of a LPP at each step. This is because, here we do not attempt to find the feasible direction that gives the best local improvement, but rather find a usable feasible direction that possibly may not be as good, but is certainly much easier to compute.

For discussing Rosen's gradient projection method, we take our NLP in the following form

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to,} \\ &&& a_{(i)}x \le b_i,\ i \in I_1 \\ &&& a_{(i)}x = b_i,\ i \in I_2\ , \end{aligned} \tag{10.6}$$

where $I = \{1, 2, \ldots, m\}$, $I_1 \cup I_2 = I$, and $a_{(i)}$ is the ith row of the coefficient matrix A.

If problem (10.6) is feasible then we find a feasible solution $x^{(0)}$ by using the Phase-I of the simplex method, and initiate the descent process of our algorithm from $x^{(0)}$. Let $x^{(k)}$ be the current feasible point.

We next identify those constraints in (10.6) which are active at $x^{(k)}$, i.e. they hold as equations at $x^{(k)}$. Let

$$I(x^{(k)}) = \{i \in I : a_{(i)}x^{(k)} = b_i\}$$

be the index set of active constraint at $x^{(k)}$. Clearly $I_2 \subset I(x^{(k)})$.

Now at the given feasible point $x^{(k)}$, we wish to find a feasible direction $d^{(k)}$ which is usable. For this, we need the direction $d^{(k)}$ to satisfy ${d^{(k)}}^T \nabla f(x^{(k)}) < 0$ so that a movement in the direction $d^{(k)}$ decreases the value of the objective function. Initially we consider directions satisfying $a_{(i)}d^{(k)} = 0, i \in I(x^{(k)})$, so that all active constraints remain active at the new point $x^{(k+1)}$ as well, where $x^{(k+1)} = x^{(k)} + \bar{\alpha}_k\, d^{(k)}$, $\bar{\alpha}_k > 0$ being the step size. This requirement amounts to requiring that $d^{(k)}$ lie on the tangent space M, which is defined by the active constraint at $x^{(k)}$. In other words, we need the projection of the chosen direction on the subspace M of the active constraints at $x^{(k)}$, to get the new point $x^{(k+1)}$. In Rosen's gradient projection method, we take $d^{(k)}$ as the projection of the negative gradient at $x^{(k)}$, i.e. $-\nabla f(x^{(k)})$, on the subspace M.

To compute this projection, we obtain the corresponding *projection matrix* P_k and take $d_k = P_k(-\nabla f(x^{(k)}))$. For this, let A_q be the matrix which is composed of the rows corresponding to active constraints at $x^{(k)}$. Thus A_q is a $(q \times n)$ matrix of rank $q < n$ (we are assuming that there are no redundant rows in A). Let M and N respectively be the null space and the range space of the matrix A_q. Then $M = \{d^{(k)} : A_q d^{(k)} = 0\}$, $N = \{A_q^T\beta, \beta \in \mathbf{R}^q\}$ and $\mathbf{R}^n = M \oplus N$. Since $-g^{(k)} \in \mathbf{R}^n$, where $g^{(k)} = \nabla(f(x^{(k)}))$, we have

$$-g^{(k)} = d^{(k)} + A_q^T\, \beta_k\ . \tag{10.7}$$

But $A_q d^{(k)} = 0$ and hence (10.7) gives

$$\beta_k = -(A_q A_q^T)^{-1} A_q\, g^{(k)}\ , \tag{10.8}$$

and

$$d^{(k)} = -[I - A_q^T(A_q A_q^T)^{-1} A_q]\, g_k\ . \tag{10.9}$$

Thus $d^{(k)} = P_k(-g^{(k)})$ where $g^{(k)} = \nabla f(x^{(k)})$ and P_k is the projection matrix given by

$$P_k = [I - (A_q^T(A_qA_q^T)^{-1}A_q] \,. \tag{10.10}$$

We can easily check that $(g^{(k)})^T d^{(k)} = ((g^{(k)})^T + (d^{(k)})^T - (d^{(k)})^T)d^{(k)} = -(d^{(k)})^T d^{(k)} = -\|d^{(k)}\|^2 < 0$, hence $d^{(k)}$ is a usable direction, provided $d^{(k)} \neq 0$. Here we have used the fact that $g^{(k)} + d^{(k)}$ is perpendicular to $d^{(k)}$ because $\mathbf{R}^n = M \oplus N$.

Next we consider the possibility that the projective negative gradient is zero, i.e. $d^{(k)} = 0$. In that case (10.7) gives

$$g^{(k)} + A_q^T\beta_k = 0 \,. \tag{10.11}$$

At this stage let us recall that A_q is composed of rows corresponding to the active constraints at $x^{(k)}$. Therefore, if the components of β_k for the active constraints are non-negative, then (10.11) implies that KKT conditions are satisfied at $x^{(k)}$ and therefore the process terminates. However, if at least one of the component of β_k (say β_{jk}) is less than zero, then it is possible to move in a new direction to get an improved point. This new direction is obtained by relaxing the inequality corresponding to that j for which $\beta_{kj} < 0$, i.e. the inequality $a_{(j)}x \leq b_j$ and get the new matrix $A_{\bar{q}}$ by deleting the row $a_{(j)}$ from A_q. The new direction is then obtained by projecting the negative gradient $-g^{(k)}$ onto the subspace determined by the rows of $A_{\bar{q}}$. Let $\bar{d}^{(k)}$ be this projection. then we can show that $(g^{(k)})^T\bar{d}^{(k)} < 0$ and $a_{(j)}\bar{d}^{(k)} < 0$. Here we have used the relations $-g^{(k)} = A_q^T\beta_k = \bar{d}^{(k)} + A_{\bar{q}}^T\bar{\beta}_k$, $\bar{d}^{(k)} \neq 0$, $(g^{(k)})^T\bar{d}^{k} < 0$, $A_{\bar{q}}\bar{d}^{(k)} = 0$ and $\beta_{jk} < 0$.

Thus $\bar{d}^{(k)}$ is a direction of descent and it is also feasible, because $a_{(i)}\bar{d}^{(k)} = 0$, $i \in I(x^{(k)})$, $(i \neq j)$ and $a_{(j)}\bar{d}^{(k)} < 0$.

We next find $\bar{\alpha}$ by finding the length of the feasible segment of the line originate at $x^{(k)}$ and then minimizing f over that segment. This seems to be the most natural way of finding $\bar{\alpha}$, but there could certainly be more efficient ways of determining the same.

Stepwise Description of the Algorithm

Step 1 Start with a feasible point $x^{(0)}$ (set $k = 0$).

Step 2 Identify, the matrix A_q and hence obtain the projection matrix P_k, given by $P_k = [I - (A_q^T(A_qA_q^T)^{-1}A_q] \,.$

Step 3 Evaluate $g^{(k)} = \nabla f(x^{(k)})$ and take $d^{(k)} = P_k(-g^{(k)})$.

Step 4 If $d^{(k)} \neq 0$, Find $\alpha_1^{(k)}$ and $\alpha_2^{(k)}$ such that

$$\alpha_1^{(k)} = \text{Max}\{\alpha \in \mathbf{R} : x^{(k)} + \alpha d^{(k)} \text{is feasible}\}$$

and

$$\alpha_2^{(k)} = \text{Min}\{f(x^{(k)} + \alpha d^{(k)}) : 0 \le \alpha \le \alpha_1^{(k)}\}.$$

Set $x^{(k+1)} = x^{(k)} + \alpha_2^{(k)} d^{(k)}$ and return to Step 2.

Step 5 If $d^{(k)} = 0$, find $\beta_k = -(A_q A_q^T)^{-1} A_q\, g^{(k)}$.

(a) If $\beta_{kj} \ge 0$ for all j corresponding to active constraints, stop as $x^{(k)}$ becomes a KKT point.

(b) If some $\beta_{kj} < 0$, then delete the jth row from A_q to construct $A_{\bar{q}}$ and return to Step 2. In practice we choose the most negative component of β_k to delete the row from A_q to construct the matrix $A_{\bar{q}}$. Further, in case A_q consists of only one row and that too has to be deleted, we take $d_k = -g^{(k)}$ and proceed as usual.

We now illustrate the working of the present algorithm.

Example 10.4.1 *Solve the following problem by Rosen's gradient projection method*

$$\begin{aligned} &\textit{Min} && (x_1 - 3)^2 + (x_2 - 7)^2 \\ &\textit{subject to} && \\ & && x_1 - 2x_2 \le 0 \\ & && x_1 + 2x_2 \le 12 \\ & && -x_1 + 6x_2 \le 24 \\ & && x_1 \ge 0, x_2 \ge 1. \end{aligned}$$

Solution Here $f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 7)^2$ and $\nabla f(x) = (-6 + 2x_1, -14 + 2x_2)^T$. Also for applying the algorithm, we need to write the given NLP in the form (10.6), and therefore the last two constraints are written as $-x_1 \le 0$ and $-x_2 \le -1$.

First Iteration

Let $x^{(0)} = (1, 1)^T$. This gives $-g^{(0)} = -\nabla f(x^{(0)}) = (4, 12)^T$. At $x^{(0)}$ only the constraint $-x_2 \le -1$ is active and hence $A_q = [0 \quad -1]$. Therefore $(A_q A_q^T)^{-1} = 1$ and $P_0 = I - A_q(A_q A_q^T)^{-1} A_q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, which gives $d^{(0)} = P_0(-g^{(0)}) = (4, 0)^T$.

As $d^{(0)} \ne 0$, we have to find $\alpha_1^{(0)}$ and $\alpha_2^{(0)}$ respectively. We recall that $\alpha_1^{(0)} = \text{Max}\{\alpha > 0 : x^{(0)} + \alpha d^{(0)} \text{is feasible}\}$. Thus

$$\alpha_1^{(0)} = \text{Max}\{\alpha > 0 : (1 + 4\alpha, 1)^T \text{is feasible}\} = 1/4.$$

Next we compute $\alpha_2^{(0)}$ by using

$$\begin{aligned} \alpha_2^{(0)} &= \text{Min}\{f(x^{(0)} + \alpha d^{(0)} : 0 < \alpha \le \alpha_1^{(0)})\} \\ &= \text{Min}\{(1 - 4\alpha - 3)^2 + (7 - 1)^2 : 0 < \alpha \le 1/4\ .\} \\ &= 1/4\ . \end{aligned}$$

This gives $x^{(1)} = x^{(0)} + \alpha_2^{(0)} d^{(0)} = (2, 1)^T$.

Here we may note that $f(x^{(0)}) = 40$ and $f(x^{(1)}) = 37$.

Second Iteration

At $x^{(1)} = (2, 1)^T$, the first and fifth constraints are active and so

$$A_q = \begin{bmatrix} 1 & -2 \\ 0 & -1 \end{bmatrix}, (A_q A_q^T)^{-1} = \begin{bmatrix} 1 & -2 \\ -2 & 5 \end{bmatrix}.$$

This gives $g^{(1)} = \nabla f(x^{(1)}) = (-2, -12)^T$, $P_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $d^{(1)} = P_1(-g^{(1)}) = (0, 0)^T$.

Now $d^{(1)}$ is zero and therefore we have to compute β_1 which is given by $\beta_1 = -(A_q A_q^T) A_q\, g^{(1)} = (2, -16)^T$. As not all components of β_1 are non-negative, $x^{(1)}$ is not a KKT point. Further, $\beta_{12} = -16 < 0$ and hence we have to drop the second row from the current A_q to get new A_q as $A_q = [1 \quad -2]$, which gives $P_1 = \begin{bmatrix} 4/5 & 2/5 \\ 2/5 & 1/5 \end{bmatrix}$ and $d^{(1)} = P_1(-g^{(1)}) = (32/5, 16/5)^T$. As $d^{(1)} \neq 0$, we again determine $\alpha_1^{(1)}$ and $\alpha_2^{(1)}$ to get $\alpha_1^{(1)} = 5/8$, $\alpha_2^{(1)} = 1/2$. This gives $x^{(2)} = x^{(1)} + \alpha_2^{(1)} d^{(1)} = (26/5, 13/5)^T$ and $f(x^{(2)}) = 121/5$.

Third Iteration

At $x^{(2)}$ only the first constraint is active and hence $A_q = [1 \quad -2]$. This gives $d^{(2)} = (0, 0)^T$ and $\beta_2 = -22/5 < 0$; and hence the first constraint should be relaxed, i.e. the first row from A_q should be dropped to get the new A_q. But A_q has only one row and that is required to be dropped and therefore we have to take $d^{(2)} = -g^{(2)} = (-22/5, -44/5)^T$, and determine $\alpha_1^{(2)}$ and $\alpha_2^{(2)}$ as before. This gives $x^{(3)} = x^{(2)} + \alpha_2^{(2)} d^{(2)} = (14/3, 11/3)^T$ and $f(x^{(3)}) = 125/9$.

Fourth Iteration

At $x^{(3)}$, the second constraint is active. Therefore $A_q = [1 \quad 2]$, $g^{(3)} = (10/3, -20/3)^T$ and $d^{(3)} = (-16/3, 8/3)^T$ giving $x^{(4)} = (3, 9/2)^T$ with $f(x^{(4)}) = 25/4$.

Fifth Iteration

At $x^{(4)} = (3, 9/2)^T$ we have $g^{(4)} = (0, -5)^T$, $A_q = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$ and $d^{(4)} = (0, 0)^T$. Also $\beta_4 = (5/8, 5/8)^T \geq 0$ and therefore $x^{(4)}$ is a KKT point. As the objective function f is a

convex function, $x^{(4)}$ is a global min point of the given problem. Hence $(x_1^* = 3,\ x_2^* = 9/2)$ is an optimal solution of the given NLP and the optimal value is $49/4$.

10.5 The Penalty Function Method: Motivation

To motivate the penalty function method, let us consider the following simple problem in one variable

$$\begin{aligned} &\text{Min} && x^2 \\ &\text{subject to} \\ & && 1 \le x \le 4. \end{aligned} \tag{10.12}$$

Here the function to be minimized is $f(x) = x^2$ and the feasible region is the interval $[1, 4]$, and hence obviously the minimizing point is $\bar{x}$=1. Now if we define a function $\widetilde{P}(x)$ as

$$\widetilde{P}(x) = \begin{cases} 0, & \text{if x is feasible (i.e. } 1 \le x \le 4) \\ +\infty, & \text{otherwise (i.e. } x > 4 \text{ or } x < 1) \end{cases}$$

and construct a new problem

$$\underset{x \in R}{\text{Min}} \quad (x^2 + \widetilde{P}(x)), \tag{10.13}$$

then problem (10.12) and (10.13) are equivalent. This is because in (10.13), as $\widetilde{P}(x) = +\infty$ for x infeasible, the minimization has to take place in the feasible region, i.e. $[1, 4]$ only. Also problem (10.13) is an unconstrained minimization problem.

The above discussion suggests that given a nonlinear programming problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \le 0, \ \ (i = 1, 2, \ldots, m), \end{aligned} \tag{10.14}$$

we can always associate an equivalent unconstrained minimization problem

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \ \ (f(x) + \widetilde{P}(x)), \tag{10.15}$$

where

$$\widetilde{P}(x) = \begin{cases} x, & \text{if } x \in S; \\ +\infty, & \text{if } x \notin S. \end{cases}$$

Here $S = \{x \in \mathbf{R}^n : g_i(x) \le 0, (i = 1, 2, \ldots, m)\}$ is the feasible region of the given (NLP)(10.14). For the sake of convenience, let the equivalent unconstrained problem (10.15) corresponding to given NLP (10.14) be denoted by $\widetilde{UMP}$.

Thus if we wish to solve the given NLP we may very well solve the equivalent unconstrained minimization problem $\widetilde{UMP}$. But can we really solve $\widetilde{UMP}$? Well, the answer has to be 'NO', because the objective function of $\widetilde{UMP}$ is not even continuous (as $\widetilde{P}(x)$ is not so) whereas most of the well known unconstrained minimization techniques are gradient based. Here we should note that the difficulty is not because $\widetilde{P}(x)$ takes the values $+\infty$ for infeasible x; the argument remains valid even if $\widetilde{P}(x)$ takes a very large finite positive value M because it still remains discontinuous.

The way $\widetilde{P}(x)$ is defined, it penalizes very heavily the decision maker for not being in the feasible region but imposes no penalty if he is in the feasible region. So it is natural to call $\widetilde{P}(x)$ a *penalty function*- in fact we shall call it an *ideal penalty function* and reserve the term penalty function for those, which are smooth penalty functions to be defined later in the sequel.

But are there smooth penalty functions? The answer is 'Yes'. These functions assign zero penalty if the point x is feasible and assign positive penalty for the point x if it is not feasible, in such a manner that the penalty becomes more and more the farther we are away from the feasible region. One typical example for problem (10.12) could be

$$P(x) = \begin{cases} (1-x)^2, & x \le 1 \\ 0, & 1 \le x \le 4 \\ (x-4)^2, & x \ge 4 \end{cases}$$

or equivalently

$$P(x) = (\text{Max}(1-x, 0))^2 + (\text{Max}(x-4, 0))^2$$

whose graph is shown in Fig 10.1.

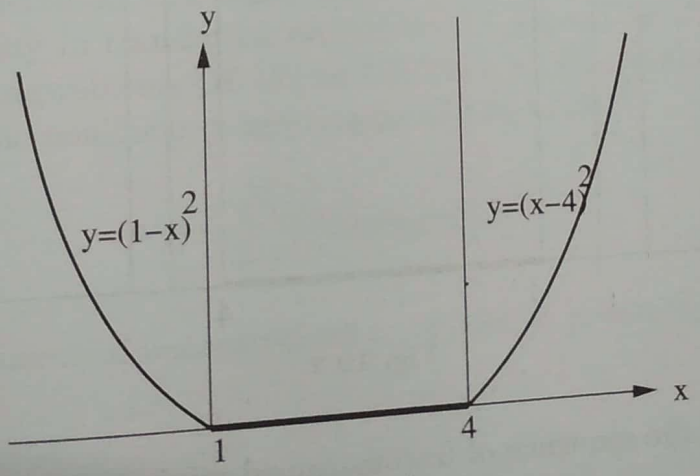


Fig. 10.1.

Here we may check that $P(x)$ is really differentiable(using chain rule) with

$$P'(x) = -2\text{Max}(1-x, 0) + 2\text{Max}(x-4, 0).$$

Thus if we wish to solve the given NLP we may very well solve the equivalent unconstrained minimization problem $\widetilde{UMP}$. But can we really solve $\widetilde{UMP}$? Well, the answer has to be 'NO', because the objective function of $\widetilde{UMP}$ is not even continuous (as $\widetilde{P}(x)$ is not so) whereas most of the well known unconstrained minimization techniques are gradient based. Here we should note that the difficulty is not because $\widetilde{P}(x)$ takes the values $+\infty$ for infeasible x; the argument remains valid even if $\widetilde{P}(x)$ takes a very large finite positive value M because it still remains discontinuous.

The way $\widetilde{P}(x)$ is defined, it penalizes very heavily the decision maker for not being in the feasible region but imposes no penalty if he is in the feasible region. So it is natural to call $\widetilde{P}(x)$ a *penalty function*- in fact we shall call it an *ideal penalty function* and reserve the term penalty function for those, which are smooth penalty functions to be defined later in the sequel.

But are there smooth penalty functions? The answer is 'Yes'. These functions assign zero penalty if the point x is feasible and assign positive penalty for the point x if it is not feasible, in such a manner that the penalty becomes more and more the farther we are away from the feasible region. One typical example for problem (10.12) could be

$$P(x) = \begin{cases} (1-x)^2, & x \leq 1 \\ 0, & 1 \leq x \leq 4 \\ (x-4)^2, & x \geq 4 \end{cases}$$

or equivalently

$$P(x) = (\text{Max}(1-x, 0))^2 + (\text{Max}(x-4, 0))^2$$

whose graph is shown in Fig 10.1.

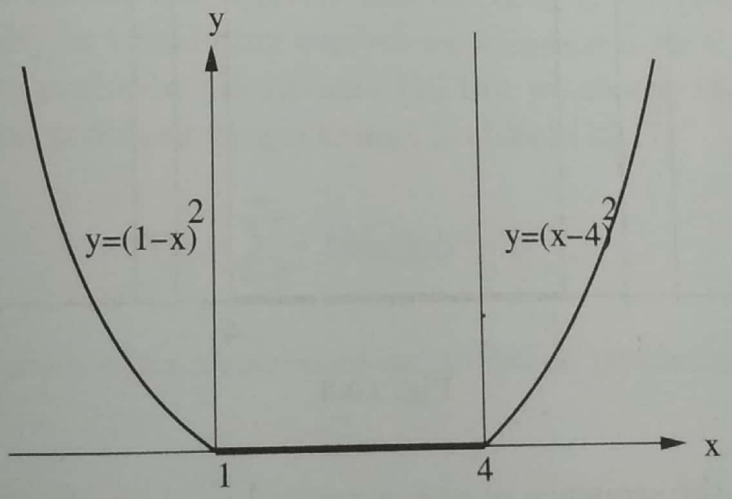


Fig. 10.1.

Here we may check that $P(x)$ is really differentiable(using chain rule) with

$$P'(x) = -2\text{Max}(1-x, 0) + 2\text{Max}(x-4, 0).$$

$\alpha \to +\infty$, $(UMP)_\alpha \to (\widetilde{UMP})$ because $\alpha P(x) \to \widetilde{P}(x)$. The important point to note here is that for each α, $(UMP)_\alpha$ is a smooth unconstrained minimization problem and therefore can be solved by the standard unconstrained minimization techniques discussed earlier in Chapter 9. If $\bar{x}(\alpha)$ is an optimal solution of $(UMP)_\alpha$, then we expect that as $\alpha \to +\infty$, $\bar{x}(\alpha)$ should converge to $\bar{x}(\alpha) = 1$, the minimizing point of problem (10.12). Infact that is going to happen as below given figure (Fig 10.4) suggests

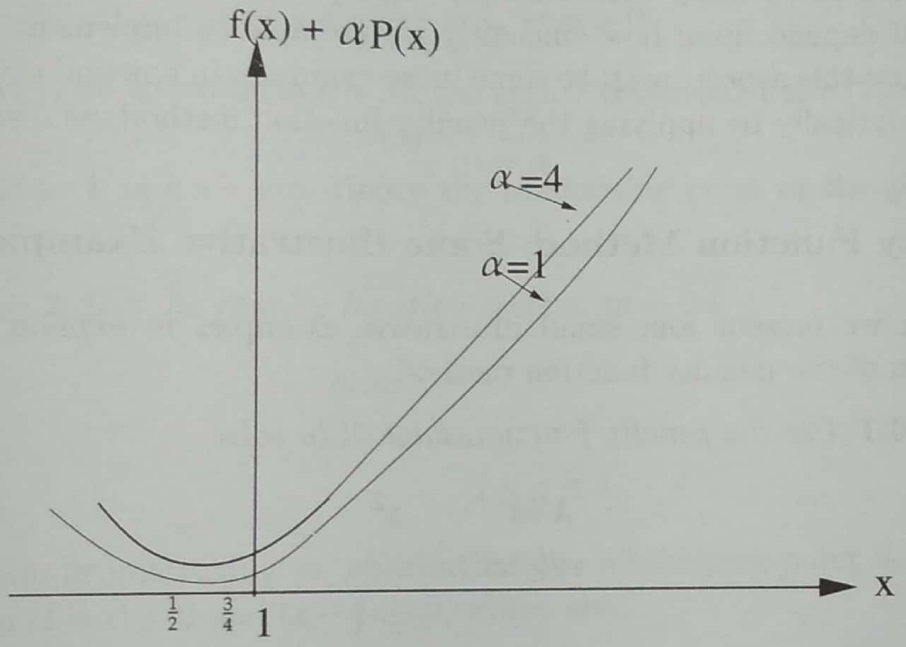


Fig. 10.4.

Once we have understood the construction of $(UMP)_\alpha$ for problem(10.12), there seems to be no difficulty in translating everything for general NLP (10.14). Therefore taking motivation from problems (10.12) and (10.16), we choose the following smooth penalty function for the nonlinear programming problem (10.17)

$$P(x) = \sum_{i=1}^{m} (\text{ Max}(g_i(x), 0))^2 \tag{10.17}$$

and construct the sequence of unconstrained optimization problems $(UMP)_\alpha$ as

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \quad f(x) + \alpha P(x)\,. \tag{10.18}$$

Let $\bar{x}(\alpha)$ be a optimal solution of problem $(UMP)_\alpha$ and $\lim_{x \to +\infty} (\bar{x}(\alpha)) = \bar{x}$. Then we expect $\bar{x}$ to be an optimal solution of NLP (10.14). This is infact true as we shall see in the sequel.

The following derivative formula will be useful in the implementation of the penalty function method.

$$\frac{\partial P}{\partial x_j} = 2\sum_{i=1}^{m} \text{Max}(g_i(x), 0)\, \frac{\partial g_i(x)}{\partial x_j}\,. \tag{10.19}$$

The above discussion suggests that problem (10.14) can possibly be solved by solving a sequence of unconstrained minimization problems $(UMP)_\alpha$ which have been constructed by using the penalty function $P(x)$. This procedure will be of some practical use or not, will depend upon how efficiently we are able to implement it numerically. Before we discuss this aspect, we take some more examples in one and two variables and solve them analytically by applying the penalty function method, as discussed above.

10.6 Penalty Function Method: Some Illustrative Examples

In this section we present four small illustrative examples to explain the analytical implementation of the penalty function method.

Example 10.6.1 *Use the penalty function method to solve*

$$\begin{aligned} &\textit{Min} && x^2 \\ &\textit{subject to} && \\ & && x \geq 1. \end{aligned}$$

Solution By sketching the graph of the function $f(x) = x^2$ over $[1, \infty]$, we obtain the minimizing point as $\bar{x} = 1$.

Now to solve the given problem by the penalty function method, we have to choose the penalty function $P(x)$. As explained in Section 10.5, we have to first express all constraints in the form $g_i(x) \leq 0$ and then take $P(x)$ as defined at (10.17).

For our example we first write the constraint $x \geq 1$ as $(1 - x) \leq 0$ and therefore $P(x) = (\text{Max}(1 - x, 0))^2$.

Next we construct the unconstrained minimization problem $(UMP)_\alpha$, $\alpha > 0$ and obtain its optimal solution $\bar{x}(\alpha)$. Then $\lim_{\alpha \to +\infty} \bar{x}(\alpha) = \bar{x}$ is going to be an optimal solution of the given optimization problem. Here, problem $(UMP)_\alpha$ is

$$\underset{x \in \mathbf{R}}{\text{Min}} \quad q(x, \alpha) = x^2 + \alpha(\text{Max}(1 - x, 0))^2.$$

Since for $\alpha > 0$, $q(x, \alpha)$ is a convex function of x and so to find its optimal solution $\bar{x}(\alpha)$, we find the derivative of $q(x, \alpha)$ w.r.t x and equate it to zero. This gives

$$x - \alpha\, \text{Max}(1 - x, 0) = 0\,, \tag{10.20}$$

which we shall called as the *basic equation of optimality.*

Now we consider two cases

Case 1 $\text{Max}(1-x,0)=0$. This gives $1-x\le 0$, i.e. $x\ge 1$. Then (10.20) gives $x=0$ which is not more than or equal to one. So this case is not possible.

Case 2. $\text{Max}(1-x,0)=(1-x)$. This gives $(1-x)\ge 0$, i.e. $x\le 1$. Also for this case (10.20) becomes

$$x-\alpha(1-x)=0,$$

i.e.

$$x=\frac{\alpha}{1+\alpha}<1, \text{ as } \alpha>0.$$

Therefore there is no contradiction with the earlier condition $x\le 1$. Thus

$$\bar{x}(\alpha)=\frac{\alpha}{1+\alpha},$$

which tends to 1 as $\alpha\to+\infty$. Hence the minimizing point of the given problem is $\bar{x}=1$.

Example 10.6.2 *Use the penalty function method to solve*

$$\begin{aligned} &\textit{Min} && x^2\\ &\textit{subject to}\\ &&& 1\le x\le 4. \end{aligned}$$

Solution Again, geometrically we obtain that the minimizing point is $\bar{x}=1$. Now the constraints are $(1-x)\le 0$ and $(x-4)\le 0$, which give

$$P(x)=(\text{Max}(1-x,0))^2+(\text{Max}(x-4,0))^2$$

and therefore for $\alpha>0$, the unconstrained minimization problem $(UMP)_\alpha$ is

$$\underset{x\in\mathbf{R}}{\text{Min}} \quad q(x,\alpha)=x^2+\alpha P(x).$$

Now equating the determinant of $q(x,\alpha)$ w.r.t x to zero, we get

$$x-\alpha\text{Max}(1-x,0)+\alpha\text{Max}(x-4,0)=0, \tag{10.21}$$

which becomes the basic equation of optimality for the given problem. Again we consider various cases and discard all those which are not possible, thereby obtaining $\bar{x}(\alpha)$ from those cases which are possible (here only one case will be possible: Why?)

Case(i) $\text{Max}(x-4,0)=(x-4)$. This implies $(x-4\ge 0)$ i.e. $x\ge 4$. But $x\ge 4$ gives $-x\le -4$, i.e. $(1-x)\le -3<0$ and hence $\text{Max}(1-x,0)=0$. Therefore, for this case, (10.21) becomes

$$x-\alpha.0+\alpha(x-4)=0,$$

i.e.

$$x(\alpha)=\frac{4\alpha}{1+\alpha}<4,$$

but this is not more than or equal to 4 as obtained above. So this case is not possible.

Case(ii) $\text{Max}(1-x,0) = 0$ and $\text{Max}(x-4,0) = 0$. These conditions imply $x \geq 1$ and $x \leq 4$. Also for these conditions, equation (10.21) becomes $x = 0$ which is not consistent with the implications $x \geq 1$ and $x \leq 4$. Therefore, this case is also not possible.
Case(iii) $\text{Max}(1-x,0) = (1-x)$. This implies $(1-x) \geq 0$ i.e. $x \leq 1$. But then $(x-4) \leq (1-4) = -3 < 0$, and hence $\text{Max}(x-4,0) = 0$. Therefore equation (10.21) gives

$$x - \alpha(1-x) = 0,$$

i.e.

$$\bar{x}(\alpha) = \frac{\alpha}{(1+\alpha)} < 1 \quad (\alpha > 0),$$

which is consistent with the implication $x \leq 1$. Therefore this is the only possible case and we see the minimizing point

$$\bar{x} = \lim_{\alpha \to +\infty} \frac{\alpha}{(1+\alpha)} = 1.$$

Remark 10.6.1 *Looking at Examples (10.6.1) and (10.6.2) above, we should not feel that penalty function method is applicable only when the minimizing point is on the boundary of the feasible region. The Example (10.6.3) given below illustrates that the penalty function method will be able to locate the minimizing point, even if it is an interior point.*

Example 10.6.3 *Use the penalty function method to solve*

$$\begin{aligned} &\textit{Min} \quad x^2 \\ &\textit{subject to} \\ &\quad -1 \leq x \leq 1. \end{aligned}$$

Solution The constraints are $(x-1) \leq 0$ and $-(1+x) \leq 0$. Therefore

$$P(x) = (\text{Max}(x-1,0))^2 + (\text{Max}(-(1+x),0))^2,$$

and problem $(UMP)_\alpha$ is

$$\min_x \; q(x,\alpha) = x^2 + \alpha P(x),$$

giving the basic equation of optimality as

$$x + \alpha \text{Max}(x-1,0) - \alpha \text{Max}(-(1+x),0) = 0 \tag{10.22}$$

For this problem, we can check that the only possibility is $\text{Max}(x-1,0) = 0$ and $\text{Max}(-(1+x),0) = 0$. This is because first equality gives $x \leq 1$ while the second equality gives $x \geq -1$, and there is no contradiction. Therefore (10.22) $x + \alpha.0 - \alpha.0 = 0$, i.e. $\bar{x}(\alpha) = 0$. Hence the minimizing point $\bar{x}$ equals $\lim_{\alpha \to +\infty} \bar{x}(\alpha)$, i.e. $\bar{x} = 0$.

Let us verify that one of the other cases, e.g. $\text{Max}(x-1,0) = (x-1)$ is not possible. If possible let $\text{Max}(x-1,0) = (x-1)$. But then this gives $x \geq 1$, i.e. $-x \leq -1$, i.e. $-1-x \leq -2 < 0$. Hence $\text{Max}(-1-x,0) = 0$. Therefore equation (10.22) gives $x + \alpha(x-1) = 0$, i.e. $\bar{x}(\alpha) = \dfrac{\alpha}{1+\alpha} < 1$, whereas the assumed condition gives $x \geq 1$. So this case is not possible.

We next take an example in two variables to understand the practical difficulties with the analytical implementation of the penalty function method.

Example 10.6.4 *Use the penalty function method to solve*

$$\begin{aligned} &\textit{Max} && 20x_1 + 16x_2 - 2x_1^2 - x_2^2 - (x_1+x_2)^2 \\ &\textit{subject to} \\ & && x_1 + x_2 \leq 5 \\ & && x_1 \geq 0,\ x_2 \geq 0. \end{aligned}$$

Solution To solve the given problem by the penalty function method we have to express the given problem as

$$\begin{aligned} &-\text{Min} && -20x_1 - 16x_2 + 3x_1^2 + 2x_2^2 + 2x_1x_2 \\ &\text{subject to} \\ & && x_1 + x_2 - 5 \leq 0 \\ & && -x_1 \leq 0 \\ & && -x_2 \leq 0. \end{aligned}$$

We observe here that the objective function to be minimized is a convex function and the constraints are linear. Now the penalty function $P(x_1,x_2)$ is given by
$P(x_1,x_2) = (\text{Max}(x_1+x_2-5,0))^2 + (\text{Max}(-x_1,0))^2 + (\text{Max}(-x_2,0))^2$,
and for $\alpha > 0$ problem $(UMP)_\alpha$ is

$$\underset{(x_1,x_2)\in \mathbf{R}^2}{\text{Min}} \quad q(x_1,x_2,\alpha) = f(x_1,x_2) + \alpha P(x_1,x_2),$$

where $f(x_1,x_2) = -20x_1 - 16x_2 + 3x_1^2 + 2x_2^2 + 2x_1x_2$, and $P(x_1,x_2) = (\text{Max}(x_1+x_2-5,0))^2 + (\text{Max}(-x_1,0))^2 + (\text{Max}(-x_2,0))^2$.

For $\alpha > 0$, $q(x_1,x_2)$ is a convex function of (x_1,x_2) and therefore to solve $(UMP)_\alpha$ we have to evaluate the partial derivatives of $q(x_1,x_2,\alpha)$ w.r.t x_1 and x_2 and equate them to zero. This gives

$$-20 + 6x_1 + 2x_2 + 2\alpha\text{Max}(x_1+x_2-5,0) - 2\alpha\text{Max}(-x_1,0) = 0 \qquad (10.23)$$

and,

$$-16 + 2x_1 + 4x_2 + 2\alpha\text{Max}(x_1+x_2-5,0) - 2\alpha\text{Max}(-x_2,0) = 0, \qquad (10.24)$$

which constitute the basic equations of optimality for the given problem.

Now we have to solve the above two equations to get $\overline{x_1}(\alpha)$ and $\overline{x_2}(\alpha)$, i.e. get x_1 as a function of α and x_2 as a function of α. As each of these equations has two 'max'

terms, we have to consider all possibilities in equations (10.23) and (10.24) to identify the cases which are possible and then solve the resulting equations. This gives

$$\overline{x_1}(\alpha) = \frac{7\alpha + 12}{3\alpha + 5},$$

and

$$\overline{x_2}(\alpha) = \frac{8\alpha + 14}{3\alpha + 5}.$$

Therefore the optimal solution of the given nonlinear programming problem is $(\overline{x_1}, \overline{x_2})$, where

$$\overline{x_1} = \lim_{\alpha \to +\infty} \overline{x_1}(\alpha) = \frac{7}{3},$$

and

$$\overline{x_2} = \lim_{\alpha \to +\infty} \overline{x_2}(\alpha) = \frac{8}{3}.$$

Remark 10.6.2 *The above example illustrates that for a general NLP, there will be n equations given by $\nabla_x q(x, \alpha) = 0$ to be solved and there will be m 'max terms' in each of these equations. Obviously in this scenario, it is almost impossible to obtain $\overline{x_1}(\alpha), \overline{x_2}(\alpha), \cdots, \overline{x_n}(\alpha)$ explicitly as a function of α and then take the limit $\alpha \to +\infty$ to get an optimal solution $(\overline{x_1}, \overline{x_2}, \cdots, \overline{x_n})$ of the given (NLP). Therefore it is imperative that we do not insist to get $\overline{x_1}, \overline{x_2}, \cdots, \overline{x_n}$ explicitly as a function of α, and think of some suitable numerical implementation. It is natural to think that any numerical implementation will require numerical values of α, but then how to interpret the statement '$\alpha \to +\infty$'. This aspect of the penalty function method we discuss in the next section.*

10.7 Numerical Implementation of the Penalty Function Method

In this section we describe the stepwise procedure for the numerical implementation of the penalty function method and prove its mathematical justification in Section 10.8.

Let us recall that the given nonlinear programming problem is in the form

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \leq 0\,, \quad (i = 1, 2, \ldots, m), \end{aligned} \tag{10.25}$$

where f and g_i $(i = 1, 2, \cdots, m)$ are differentiable convex functions.

Step 1. Choose a suitable penalty function $P(x)$. In practice we take the penalty function as $P(x) = \sum_{i=1}^{m} (\text{Max}(g_i(x), 0))^2$.

Step 2. Choose an increasing sequence of positive real numbers which tends to infinity, i.e. a sequence $\{\alpha_k\}_{k=1}^{\infty}$ such that for each k, $\alpha_k > 0$, $\alpha_{k+1} > \alpha_k$ and $\{\alpha_k\} \to +\infty$. In practice we take $\alpha_1 = 1$, $\alpha_2 = 10$, $\alpha_3 = 100$, $\alpha_4 = 1000$ etc.

Step 3. Choose an arbitrary point $x^{(0)} \in \mathbf{R}^n$. Construct the following unconstrained minimization problem $(UMP)_{\alpha_1}$

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \quad q(x, \alpha_1) = f(x) + \alpha_1 P(x),$$

and solve it by a suitable unconstrained minimization technique starting with the point $x^{(0)}$. Let $\bar{x}^{(1)}$ be an optimal solution of $(UMP)_{\alpha_1}$. Set $k = 1$.

Step 4. Construct the following unconstrained minimization problem $(UMP)_{\alpha_{k+1}}$

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \quad q(x, \alpha_{k+1}) = f(x) + \alpha_{k+1} P(x).$$

Solve it by a suitable unconstrained minimization technique starting with the point $\bar{x}^{(k)}$, where $\bar{x}^{(k)}$ is the optimal solution of $(UMP)_{\alpha_k}$ as obtained at the preceding step. Thus $(UMP)_{\alpha_2}$ is solved starting with the point $\bar{x}^{(1)}$ which is obtained by solving $(UMP)_{\alpha_1}$.

Step 5. Continue till $\alpha_k P(\bar{x}^{(k)})$ is close to zero or equivalently $f(\bar{x}^{(k)})$ is close to $q(\bar{x}^{(k)}, \alpha_k)$, i.e. $\alpha_k P(\bar{x}^{(k)}) < \epsilon$ or $q(\bar{x}^{(k)}, \alpha_k) - f(\bar{x}^{(k)}) < \epsilon$ for a tolerance $\epsilon > 0$.

Remark 10.7.1 *Although we have assumed from the very beginning that the given nonlinear programming problem is a convex programming problem, the basic logic of penalty function method is valid even for nonconvex problems provided we have methods to solve those UMP's which are not necessarily convex. Since most of standard methods for UMP's do require convexity to give global optimal solution, it becomes imperative to assume that the nonlinear programming problem is a convex programming problem. But theoretically at least, penalty function method is valid for nonconvex NLP's as well.*

Example 10.7.1 *Use the numerical implementation of the penalty function method to solve*

$$\begin{aligned} &\textit{Min} && (x_1 - 2)^4 + (x_1 - 2x_2)^2 \\ &\textit{subject to} \\ & && x_1^2 - x_2 = 0; \end{aligned}$$

starting with the point $x^{(0)} = (2, 1)^T$.

Solution The constraint $x_1^2 - x_2 = 0$ can be written as $x_1^2 - x_2 \leq 0$ and $-x_1^2 + x_2 \leq 0$. Therefore the penalty function is

$$\begin{aligned} P(x_1, x_2) &= (\text{Max}(x_1^2 - x_2, 0))^2 + (\text{Max}(-(x_1^2 - x_2), 0))^2 \\ &= (x_1^2 - x_2)^2 \end{aligned}$$

(In general for an equality constraint $g(x) = 0$, the penalty function is $P(x) = (g(x))^2$. This follows directly by writing $g(x) = 0$ as $g(x) \leq 0$ and $-g(x) \leq 0$).

Let us choose the sequence $\{\alpha_k\}$ as $\alpha_1 = 0.1$, $\alpha_2 = 1$, $\alpha_3 = 10$, $\alpha_4 = 100$ etc. and construct the unconstrained minimization problem $(UMP)_{\alpha_k}$. Then following the steps described above we obtain the below given table

k	α_k	$x^{(k)}$	$f(x^{(k)})$	$q(x^{(k)}, \alpha_k)$	$P(x^{(k)})$	$\alpha_k P(x^{(k)})$
1	0.1	(1.4539,0.7608)	0.0935	0.2766	1.8307	.1831
2	1	(1.1687,0.7407)	0.5753	0.9661	0.3908	.3908
3	10	(0.9906,0.8425)	1.5203	1.7129	0.01926	.1926
4	100	(0.9507,0.8875)	1.8917	1.9184	0.000267	.0267
5	1000	(0.9461,0.8934)	1.9405	1.9433	0.0000028	.0028

As per our stopping criterion we stop here because $f(x^{(k)}) \simeq q(x^{(k)}, \alpha_k)$ or equivalently $\alpha_k P(x^{(k)})$ is close to zero. Therefore an optimal solution of the given (NLP) can be taken as $(\overline{x_1} = 0.9461, \overline{x_2} = 0.8934)$ with the optimal value as 1.9405.

10.8 Mathematical Justification of the Penalty Function Method

In this section we establish the convergence of the penalty function method. For this, we first prove the following two lemmas.

Lemma 10.8.1. *Let $\overline{x}^{(k)}$ denote the optimal solution of $(UMP)_{\alpha_k}$, i.e. $q(\overline{x}^{(k)}, \alpha_k) = \underset{x \in \mathbf{R}^n}{Min}\ q(x, \alpha_k)$, where $q(x, \alpha_k) = f(x) + \alpha_k P(x)$, $\alpha_k > 0$. Then*

(i) $q(\overline{x}^{(k)}, \alpha_k) \leq q(\overline{x}^{(k+1)}, \alpha_{k+1})$

(ii) $P(\overline{x}^{(k)}) \geq P(\overline{x}^{(k+1)})$

(iii) $f(\overline{x}^{(k)}) \leq f(\overline{x}^{(k+1)})$.

Proof.

(i) We have

$$\begin{aligned} q(\overline{x}^{(k+1)}, \alpha_{k+1}) &= f(\overline{x}^{(k+1)}) + \alpha_{k+1} P(\overline{x}^{(k+1)}) \\ &\geq f(\overline{x}^{(k+1)}) + \alpha_k P(\overline{x}^{(k+1)}) \\ &\geq f(\overline{x}^{(k)}) + \alpha_k P(\overline{x}^{(k)}) = q(\overline{x}^{(k)}, \alpha_k), \end{aligned}$$

where the first inequality follows because $\alpha_{k+1} > \alpha_k$, while the second inequality follows because $\overline{x}^{(k)}$ is optimal to $\underset{x \in \mathbf{R}^n}{Min}\ q(x, \alpha_k)$.

(ii) By the definition of $\overline{x}^{(k)}$

$$f(\overline{x}^{(k)}) + \alpha_k P(\overline{x}^{(k)}) \leq f(\overline{x}^{(k+1)}) + \alpha_k P(\overline{x}^{(k+1)}) . \tag{10.26}$$

Similarly by the definition of $\bar{x}^{(k+1)}$,

$$f(\bar{x}^{(k+1)}) + \alpha_{k+1}P(\bar{x}^{(k+1)}) \leq f(\bar{x}^{(k)}) + \alpha_{k+1}P(\bar{x}^{(k)}) . \tag{10.27}$$

Now adding (10.26) and (10.27) we get

$$(\alpha_{k+1} - \alpha_k)P(\bar{x}^{(k+1)}) \leq (\alpha_{k+1} - \alpha_k)P(\bar{x}^{(k)}),$$

i.e.

$$P(\bar{x}^{(k+1)}) \leq P(\bar{x}^{(k)}) .$$

(iii) Again by the definition of $\bar{x}^{(k)}$,

$$f(\bar{x}^{(k+1)}) + \alpha_k P(\bar{x}^{(k+1)}) \geq f(\bar{x}^{(k)}) + \alpha_k P(\bar{x}^{(k)}) .$$

But $P(\bar{x}^{(k)}) \geq P(\bar{x}^{(k+1)})$, and therefore from the above inequality

$$f(\bar{x}^{(k+1)}) + \alpha_k P(\bar{x}^{(k+1)}) \geq f(\bar{x}^{(k)}) + \alpha_k P(\bar{x}^{(k+1)})$$

i.e

$$f(\bar{x}^{(k)}) \leq f(\bar{x}^{(k+1)}).$$

Lemma 10.8.2. *Let $\bar{x}$ be an optimal solution of the given nonlinear programming problem (10.25). Then for each k,*

$$f(\bar{x}) \geq q(\bar{x}^{(k)}, \alpha_k) \geq f(\bar{x}^{(k)}) .$$

Proof. Since $\bar{x}$ is optimal to problem (10.25), it is certainly feasible. Hence by the definition of penalty function $P(x)$, we have $P(\bar{x}) = 0$, which gives

$$f(\bar{x}) = f(\bar{x}) + \alpha_k P(\bar{x}).$$

But $\bar{x}^{(k)}$ is an optimal solution of $(UMP)_{\alpha_k}$ and therefore

$$f(\bar{x}) + \alpha_k P(\bar{x}) \geq f(\bar{x}^{(k)}) + \alpha_k P(\bar{x}^{(k)}).$$

Hence

$$f(\bar{x}) \geq f(\bar{x}^{(k)}) + \alpha_k P(\bar{x}^{(k)}). \tag{10.28}$$

As $\alpha_k P(\bar{x}^{(k)}) \geq 0$, (10.28) gives

$$f(\bar{x}) \geq q(\bar{x}^{(k)}, \alpha_k) \geq f(\bar{x}^{(k)}) .$$

Theorem 10.8.1 *Let $\{\bar{x}^{(k)}\}$, be a sequence generated by the penalty function method. Then any limit point of the sequence is an optimal solution of problem (10.25).*

Proof. Suppose that $\{\overline{x}^{(k)}\}$, $k \in \mathcal{K}$ is a convergent subsequence of $\{\overline{x}^{(k)}\}$ having limit $\hat{x}$. Then by the continuity of f we have

$$\lim_{k\in\mathcal{K}} f(\overline{x}^{(k)}) = f(\hat{x}).$$

Let $\overline{f}$ be the optimal value of the nonlinear programming problem (10.25). Then according to Lemmas 10.8.1 and 10.8.2, the sequence of values $q(\overline{x}^{(k)}, \alpha_k)$ is non-decreasing and bounded above by $\overline{f} = f(\overline{x})$. Thus

$$\lim_{k\in\mathcal{K}} q(\overline{x}^{(k)}, \alpha_k) = \overline{q} \leq \overline{f} = f(\overline{x}) .$$

Now

$$\lim_{k\in\mathcal{K}} q(\overline{x}^{(k)}, \alpha_k) - \lim_{k\in\mathcal{K}} f(\overline{x}^{(k)}) = \overline{q} - f(\hat{x}) ,$$

i.e.

$$\lim_{k\in\mathcal{K}} \alpha_k P(\overline{x}^{(k)}) = \overline{q} - f(\hat{x}) .$$

Since $P(\overline{x}^{(k)}) \geq 0$ and $\alpha_k \to +\infty$, the above implies

$$\lim_{k\in\mathcal{K}} P(\overline{x}^{(k)}) = 0 ,$$

which by the continuity of P, gives $P(\hat{x}) = 0$. We have therefore shown that $\hat{x}$ is feasible to problem (10.25).

To show that $\hat{x}$ is optimal to problem (10.25), we use Lemma 10.8.2 and note that $f(\overline{x}^{(k)}) \leq \overline{f}$, and hence

$$f(\hat{x}) = \lim_{k\in\mathcal{K}} f(\overline{x}^{(k)}) \leq \overline{f} .$$

10.9 The Barrier Function Method: Motivation

In the penalty function method we are positively penalized for being outside the feasible region and therefore we are forced to come to the feasible region and do the actual optimization. Thus in this method our movement in general is from outside to inside of the feasible region.

We may think of some sort of a converse strategy now; i.e. we start from a point which is in the interior of feasible region and introduce a function which acts like a barrier (so appropriately called a *barrier function*) so that during optimization we do not go outside the feasible region. Therefore here again we solve the given nonlinear programming problem by solving a sequence of unconstrained minimization problems, so barrier function method also falls within the general class of SUMT.

To motivate the barrier function method, we consider the following example

$$\begin{aligned} \text{Min} \quad & \frac{x}{2} \\ \text{subject to} \quad & (1-x) \leq 0 \end{aligned}$$

and check graphically that the minimizing point is $\bar{x} = 1$.

Let $S = \{x \in \mathbf{R} : (1-x) \leq 0\}$ denote the feasible region of the given problem. We now introduce a function $B : int\ S \rightarrow \mathbf{R}$ given by

$$B(x) = \frac{-1}{(1-x)},$$

where 'int S', denotes the interior of the set S, $\text{int}S = \{x : 1 < x < \infty\}$. For this function we note that (i) $B(x) \geq 0$ for all $x \in int\ S$, (ii) $B(x)$ is differentiable in its domain and (iii) $B(x) \rightarrow +\infty$ as x approaches the boundary of S.

We next consider the following unconstrained minimization problem $(UMP)_r$ for $r > 0$

$$\underset{x \in int\ S}{\text{Min}} C(x,r) = \frac{x}{2} + r\left(\frac{-1}{1-x}\right). \tag{10.29}$$

Looking at the above problem, the first reaction is about its notation; there is apparently a constraint, namely '$x \in int\ S$' and yet we are calling it an unconstrained minimization problem. The reason for this stems from the property (iii) of $B(x)$. As $B(x) \rightarrow +\infty$ whenever x approaches to boundary of S, it is obvious that starting from a point in the interior of the set S and following a descent strategy, the optimal situation of $(UMP)_r$ will automatically be in the interior of S, thereby making the constraint '$x \in int\ S$' irrelevant. If we now plot the function $C(x,r)$ for various values of r we get the curves as shown in Fig 10.5.

From Fig 10.5, we feel intuitively that if we obtain the optimal solution of $(UMP)_r$ (say $\bar{x}(r)$) then $\bar{x}(r)$ should tend to the minimizing point $\bar{x}$ as $r \rightarrow 0$. This is in fact true as we can verify for the above example by employing usual calculus; because $\frac{dC}{dx} = 0$ gives $x = 1 - \sqrt{2}r$ and $x = 1 + \sqrt{2}r$, and $\frac{d^2C}{dx^2} > 0$ for $\bar{x}(r) = 1 - \sqrt{2}r$. Thus the minimum of problem $(UMP)_r$ is attained for $\bar{x}(r) = 1 - \sqrt{2}r$ which tends to $\bar{x} = 1$ (the actual minimizing point of the given optimization problem) as $r \rightarrow 0$.

10.10 Analytical Implementation of the Barrier Function Method

Let the given nonlinear programming problem be

$$\begin{aligned} \text{Min} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad (i = 1, 2, \ldots, m), \end{aligned} \tag{10.30}$$

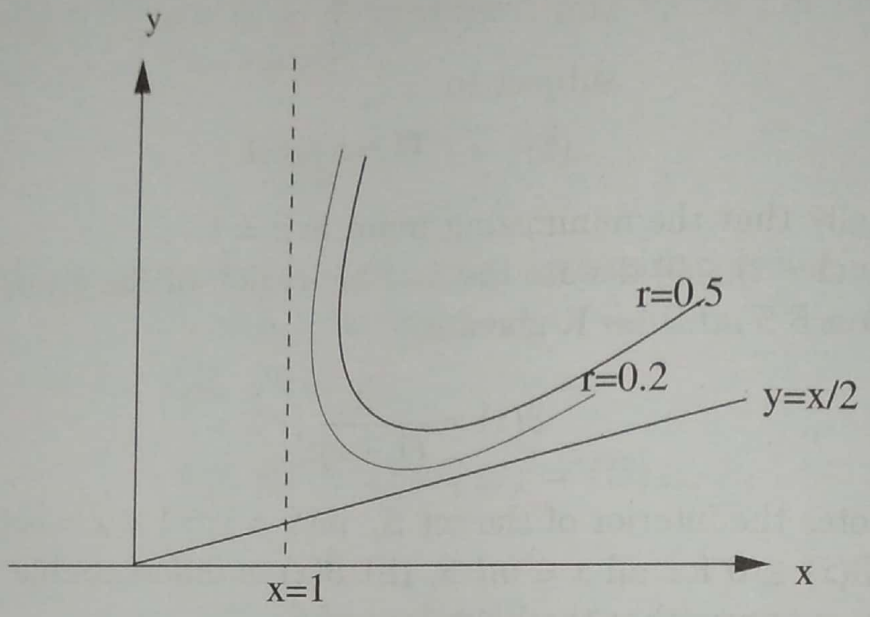


Fig. 10.5.

with its feasible region $S = \{x \in \mathbf{R}^n : g_i(x) \leq 0,\ i = 1, 2, \cdots, m\}$. We assume that $int\ S \neq \phi$. A consequence of this assumption is that problem (10.30) can not have any equality constraint.

As mentioned in the last section, the barrier function method works by establishing a barrier on the boundary of the feasible region that prevents a search procedure from leaving the feasible region. We now give a formal definition of a barrier function.

Definition 10.10.1. ***(Barrier Function)*** *A function $B : int\ S \to \mathbf{R}$ is called a barrier function if*

(i) $B(x) \geq 0$ for all $x \in int\ S$

(ii) $B(x)$ is differentiable

(iii) $B(x) \to +\infty$ as x approaches to boundary of S.

Though there could be many choices of barrier function, a typical barrier function used most often is

$$B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)}, \quad x \in int\ S\ .$$

This is indeed a barrier function, can be checked very easily as we can show that it satisfies all conditions of Definition 10.10.1. Also if each g_i is a convex function then $B(x)$ is a convex function, because $g_i(x) < 0$ for $x \in$ Int S.

Now for $r > 0$ we construct problem $(UMP)_r$ given by

$$\underset{x \in int\ S}{\text{Min}}\ C(x, r) = f(x) + rB(x). \tag{10.31}$$

This problem is certainly a constrained problem and the constraint $x \in int\ S$ seems to be more complicated than the original constraints of (10.30), namely $x \in S$. But

surprisingly this problem can be solved by using unconstrained minimization techniques and hence the notation $(UMP)_r$. This is because to find a solution of $(UMP)_r$ we start from an initial point which is in the interior of the set S and then search from that point, using steepest descent method or some other iterative descent method applicable to UMP's. As $B(x) \to +\infty$ for points x near the boundary of S, the optimal solution $\bar{x}(r)$ of $(UMP)_r$ is bound to be in *int* S and so the constraint $x \in$ *int* S need not be considered explicitly.

Once $\bar{x}(r)$ is known explicitly as function of r, we obtain $\bar{x} = \lim_{r \to 0} \bar{x}(r)$ as an optimal solution of problem (10.30).

Example 10.10.1 *Use analytical implementation of the barrier function method to solve*

$$\begin{aligned} &\textit{Min} && x^2 \\ &\textit{subject to} \\ & && -1 \le x \le 1. \end{aligned}$$

Solution Here $f(x) = x^2$ and $g_1(x) = (x-1) \le 0$ and $g_2(x) = -(1+x) \le 0$ are two constraints. Therefore we take

$$B(x) = \frac{-1}{(x-1)} - \frac{1}{-(1+x)} = \frac{-1}{(x-1)} + \frac{1}{(1+x)},$$

and introduce the following problem $(UMP)_r$ for $r > 0$

$$\underset{-1<x<1}{\text{Min}} \quad C(x,r) = x^2 - \frac{r}{(x-1)} + \frac{r}{(x+1)}.$$

At this stage, we should note that for this problem, $-1 < x < 1$, is the constraint $x \in$ *int* S which is never enforced because it follows automatically that the optimal solution of $(UMP)_r$ will be in the interior of the set S. Therefore for all practical purposes, $(UMP)_r$ is an unconstrained optimization problem. Now $\dfrac{dC}{dx} = 0$ gives

$$x\left[\frac{2r}{(x-1)^2(x+1)^2} + 1\right] = 0,$$

i.e. $\bar{x}(r) = 0$. Also $\dfrac{d^2C}{dx^2} > 0$ for $\bar{x}(r) = 0$. Therefore $\bar{x}(r) = 0$ is the optimal solution of $(UPM)_r$ and $\lim_{r \to 0} \bar{x}(r)$, i.e. $\bar{x} = 0$ is the minimizing point of the given optimization problem.

10.11 Numerical Implementation of the Barrier Function Method

The numerical implementation of the barrier function method is very similar to that of the penalty function method, which we describe in the following.

Step 1. Choose a suitable barrier function $B(x)$. In practice we take

$$B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)}.$$

Step 2. Choose a decreasing sequence of positive real numbers which tends to zero, i.e. a sequence $\{r_k\}_{k=1}^{\infty}$ such that for all k, $r_k > 0$, $r_{k+1} < r_k$ and $\{r_k\} \to 0$ as $k \to +\infty$. In practice we take $r_1 = 10$, $r_2 = 1$, $r_3 = 0.1$, $r_4 = .01$ etc.

Step 3. Choose a starting point $x^{(0)} \in$ int S and construct the unconstrained minimization problem $(UMP)_{r_1}$

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \; C(x, r_1) = f(x) + r_1 B(x) \, .$$

Next, solve $(UMP)_{r_1}$ by a suitable unconstrained minimization technique starting with the point $x^{(0)}$. Let $\overline{x}^{(1)}$ be an optimal solution of $(UMP)_{r_1}$. Set $k = 1$ (here we may note one important change in $(UMP)_{r_1}$. We write $x \in \mathbf{R}^n$ rather than $x \in$ the interior of the set S, i.e. we take the problem as unconstrained minimization problem. As has been earlier explained, this does not change the original problem as $\overline{x}^{(1)}$ will automatically be in int S due to the presence of the barrier function $B(x)$).

Step 4. Construct the following unconstrained minimization problem $(UMP)_{r_{k+1}}$

$$\underset{x \in \mathbf{R}^n}{\text{Min}} \; C(x, r_{k+1}) = f(x) + r_{k+1} B(x) \, ,$$

and solve $(UMP)_{r_{k+1}}$ by a suitable unconstrained minimization technique starting with $\overline{x}^{(k)}$, where $\overline{x}^{(k)}$ is the solution of $(UMP)_{r_k}$ as obtained at the preceding step.

Step 5. Continue till $r_k B(\overline{x}^{(k)})$ is close to zero or equivalently $f(\overline{x}^{(k)})$ is close to $C(\overline{x}^{(k)}, r_k)$; i.e. $r_k B(\overline{x}^{(k)}) < \epsilon$ or $C(\overline{x}^{(k)}, r_k) - f(\overline{x}^{(k)}) < \epsilon$ for a tolerance $\epsilon > 0$.

Remark 10.11.1 *This remark is similar to Remark 10.7.1, in the sense that we can solve even nonconvex NLP's by barrier function method provided we have procedures to obtain the global optimal solution of (nonconvex) unconstrained minimization problems* $(UMP)_{r_k}$. *Thus the convexity assumption on the objective and constraint functions is essentially used to get* $(UMP)_{r_k}$ *as convex optimization problems for each* $r_k > 0$, *so that these UMP's can be solved efficiently.*

Example 10.11.1 *Use numerical implementation of the barrier function method to solve*

$$\textit{Min} \quad (x_1 - 2)^4 + (x_1 - 2x_2)^2$$

subject to

$$x_1^2 - x_2 \leq 0.$$

Solution Before we start solving this problem, we may note that we cannot use the barrier function method if the constraint is given in the form $x_1^2 - x_2 = 0$ because one of the basic assumption here is that *int* $S \neq \phi$. Also we cannot take $x^{(0)} = (1,1)^T$ even though it is feasible, because the starting point $x^{(0)}$ has to be in the interior of the set S.

We now choose the barrier function $B(x)$ as

$$B(x) = \frac{-1}{(x_1^2 - x_2)}.$$

and take $r_1 = 10$, $r_2 = .1$, $r_3 = .01$, $r_4 = .001$ etc. Next we consider problem $(UMP)_{r_1}$ as

$$\text{Min} \quad C(x, r_1) = (x_1 - 2)^2 + (x_1 - 2x_2)^2 + 10\frac{-1}{(x_1^2 - x_2)}$$

and solve the same by a suitable unconstrained minimization technique starting with the point $x^{(0)} = (0,1)^T$. This gives an optimal solution of $(UMP)_{r_1}$ as $x^{(1)} = (.7079, 1.5318)^T$. We then construct problem $(UMP)_{r_2}$ and solve the same starting with $x^{(1)}$. We summarize these results in the following table

k	r_k	$x^{(k)}$	$f(x^{(k)})$	$C(x^{(k)}, r_k)$	$B(x^{(k)})$	$r_k B(x^{(k)})$
1	10	(0.7079,1.5315)	8.3338	18.0388	9.7105	.9705
2	1	(0.8282,1.1098)	3.8214	6.1805	2.3591	2.3591
3	0.1	(0.8989,1.9638)	2.5282	3.1701	6.4194	0.6419
4	0.01	(0.9294,0.9162)	2.1291	2.3199	19.0785	0.1907
5	0.001	(0.9403,0.9011)	2.0039	2.0629	59.0461	0.0590
6	0.0001	(0.9438,0.8966)	1.9645	1.9829	184.4451	0.0018

As per our stopping criterion we stop here because $f(x^{(k)}) \simeq C(x^{(k)}, r_k)$ or equivalently $r_k B(x^{(k)})$ is small (close to zero). Therefore an optimal solution to of the given problem can be taken as $(x_1^* = 0.9438, x_2^* = 0.8966)$ with the optimal value as 1.9645.

10.12 Mathematical Justification of the Barrier Function Method

In this section we state convergence results for the barrier function method. We do not prove these results here as they follow similar to those presented in Section 10.8 for the penalty function method.

Lemma 10.12.1. *Let* $\bar{x}^{(k)}$ *denote the optimal solution of* $(UMP)_{r_k}$, *i.e.* $C(\bar{x}^{(k)}, r_k) = \underset{x \in int\ S}{Min}\ C(x, r_k)$, *where* $C(x, r_k) = f(x) + r_k B(x)$, $r_k > 0$. *Then*

(i) $C(\bar{x}^{(k)}, r_k) \geq C(\bar{x}^{(k+1)}, r_{k+1})$

(ii) $B(\bar{x}^{(k)}) \leq B(\bar{x}^{(k+1)})$

(iii) $f(\bar{x}^{(k)}) \geq f(\bar{x}^{(k+1)})$.

Theorem 10.12.1 *Let $\{\bar{x}^{(k)}\}$ be a sequence generated by the barrier function method. Then any limit point of the sequence is an optimal solution of the given nonlinear programming problem.*

10.13 Starting Point for the Barrier Function Method

We know that to employ the barrier function method we have to start from an initial point $x^{(0)} \in int\ S$. If the problem is small then there is a possibility that such a point can be obtained by some trial and error method. However if there are many constraints (i.e. m is large) and/or many variables (i.e. n is large) then such an approach is neither possible nor desirable. Therefore, in the following we present a simple but logical way of finding a point $x^{(0)}$ in the interior of the feasible region.

Let $S = \{x \in \mathbf{R}^n : \ g_i(x) \leq 0, \ \ i = 1, 2, \cdots, m\}$ be the feasible region. Then our aim is to find $x^{(0)} \in int\ S$, where

$$int\ S = \{x \in \mathbf{R}^n : \ g_i(x) < 0, \ \ i = 1, 2, \cdots, m\}\ .$$

Let us follow the following steps

Step 1. Choose $\hat{x} \in \mathbf{R}^n$ arbitrary.

Step 2. Let $J(\hat{x}) = \{i \in I : \ g_i(x) < 0\}$, where $I = \{1, 2, \cdots, m\}$. Thus $J(\hat{x})$ is the index set of those constraints which are not active at $\hat{x}$, i.e. $J(\hat{x}) = I - I(\hat{x})$, $I(\hat{x})$ being the index set of active constraints at $\hat{x}$.

Step 3. If $J(\hat{x}) = I$, stop as $\hat{x} \in int\ S$, as $g_i(\hat{x}) < 0$ for all $i \in I$. However if $J(\hat{x}) \subset I$, go to Step 4.

Step 4. Choose $j \notin J(\hat{x})$, i.e. choose j for which $g_j(\hat{x}) \geq 0$ and construct the following problem (P_j)

$$\begin{aligned} &\text{Min} && g_j(x) \\ &\text{subject to} \\ & && g_i(x) \leq 0, \ \ i \in J(\hat{x}). \end{aligned}$$

Note that for the above problem we have a point, namely $\hat{x}$, such that $\hat{x} \in int\ S_1$, where S_1 is the feasible region of problem (P_j). Therefore problem (P_j) can be solved by the barrier function method itself by taking the starting solution as $\hat{x}$. Let $x^{(1)}$ be the optimal solution of (P_1). If $g_j(x^{(1)}) \geq 0$, Stop as $int\ S = \phi$, otherwise go to Step 2.

Step 5 Find $J(x^{(1)})$ and repeat.

It is not difficult to visualize that after at most m iterations we shall either be able to conclude that $int\ S = \phi$ or get a point $x^{(k)}$ such that $x^{(k)} \in int\ S$.

10.14 Algorithms and Their Global Convergence

Let s recall that an algorithm is said to be *globally convergent* if for an *arbitrary starting point*, the algorithm is guaranteed to generate a sequence of points converging to a point of the *solution set*. The meaning of 'arbitrary starting point' and the 'solution set' may change from one class of problems to an other class of problems. Thus for UMP's, arbitrary starting point may mean $x^{(0)} \in \mathbf{R^n}$ and solution may mean a point $\bar{x}$ such that $\|\nabla f(\bar{x})\| \leq \epsilon$. For NLP's, an arbitrary starting point may mean a feasible point $x^{(0)}$ (i.e. $x^{(0)} \in S$) and solution may mean a point $\bar{x}$ which is a KKT point. Obviously not all algorithms are globally convergent and even if an algorithm is globally convergent, establishing this property is not a simple task. These proofs depend heavily on various results from analysis and linear algebra. Also the proof of global convergence is generally algorithm dependent. This makes the analysis of the algorithms rather involved and therefore above the reach of most of the common readers. In this regard, W. Zangwill [173] in 1969 gave a unified theory of global convergence which could be used to establish the global convergence property of most of the algorithms. Our aim in this section is to give a very brief account of this important development.

Let the given optimization problem be to minimize $f(x)$ over $x \in S$, where $S \subset \mathbf{R}^n$ is the feasible region and f is the objective function. By an *algorithm* for solving this optimization problem we generally mean an *iterative process* that (i) starts from a given point $x^{(0)}$, (ii) generates the current point $x^{(k)}$ according to some prescribed set of instructions and (iii) stops as per specified stopping rule.

Mathematically, we can view an algorithm as a *point to set map* A, which for a given point $x^{(k)}$ generates a new point $x^{(k+1)} \in A(x^{(k)})$. This point to set map is called the *algorithmic map* which generates the points $x^{(0)}, x^{(1)}, \ldots, x^{(k)}, x^{(k+1)}, \ldots$ where $x^{(k+1)} \in A(x^{(k)})$ for each k. Further obtaining $x^{(k+1)}$ from $x^{(k)}$ by using the given map A is called an *iteration*.

As an example let us consider the problem of minimizing x^2 subject to $x \geq 1$. Also let $A_1(x) = \frac{1}{2}(x+1)$ and $A_2(x) = [1, \frac{1}{2}(x+1)]$, for $x \geq 1$ and $A_2(x) = [\frac{1}{2}(x+1), 1]$, for $x <$ 1; be two algorithmic maps for the given problem. Then with $x^{(0)} = 4$, the map A_1 generates the sequence {4,2.5,1.75,1.375,...} which converges to the solution $\bar{x} = 1$. Infact it converges to $\bar{x} = 1$ with any arbitrary point $x^{(0)} \in S$.

For the map A_2, the image of any point $x^{(k)}$ is a closed interval, and any point in that interval could be taken as the new point $x^{(k+1)}$. In particular with $x^{(0)} = 4$, the sequence $\{4, 2, 1.2, 1.1, 1.0, \ldots\}$ is a possible result of the algorithm A_2. Obviously many other sequences are possible for the starting point $x^{(k+1)}$ in the closed interval $A_2(x^{(k)})$ given by the map A_2.

Definition 10.14.1 (Closed Map). *Let X and Y be two nonempty closed sets in $\mathbf{R}^p$ and $\mathbf{R}^q$ respectively. Let $A : X \to Y$ be a point to set map and $x \in X$. Then the map A is said to be closed at x if*

$$x^{(k)} \in X, x^{(k)} \to x, y^{(k)} \in A(x^{(k)}), y^{(k)} \to y \Rightarrow y \in A(x).$$

If for $Z \subset X$, the map A is closed at each point in Z then we say that A is closed on Z.

For the optimization problem 'minimizing x^2 subject to $x \geq 1$' the map

$$A_3(x) = \begin{cases} [\frac{3}{2} + \frac{1}{4}x, 1 + \frac{1}{2}x], & x \geq 2 \\ \frac{1}{2}(x+1), & x < 2 \end{cases}$$

is not closed at $x = 2$, whereas A_1 and A_2 are closed everywhere.

Definition 10.14.2 (Descent Function). *Let $\Omega \subset X$ be the solution set of the given optimization problem. Then a function $\alpha : X \to X$ is called a descent function if $\alpha(y) < \alpha(x)$ for $x \notin \Omega$ and $y \in A(x)$.*

Theorem 10.14.1 (Zangwill's Global Convergence Theorem).
Let $X \subset \mathbf{R}^n$, $(X \neq \phi)$, be closed and $\Omega \subset X$, $(\Omega \neq \phi)$ be the solution set. Let $A : X \to X$ be the given algorithmic (point to set) map which is closed on the complement of Ω. Given $x^{(0)} \in X$, let the algorithm generate a sequence $\{x^{(0)}, x^{(1)}, \ldots x^{(k)}, \ldots\}$, $x^{(k+1)} \in A(x^{(k)})$, which is contained in a closed and bounded subset of X. Also let there exists a descent function α. Then either the algorithm stops in a finite number of steps to a point $\bar{x}$ in Ω or it generates a sequence $\{x^{(k)}\}$ such that every accumulation point of this sequence is a point in Ω.

Remark 10.14.1 *Most algorithms discussed in Chapters 9 and also those discussed in this chapter, are of type $A = MD$, where the map D finds the direction $d^{(k)}$ and the map M finds the optimal step sized $\overline{\alpha_k}$ for a given $d^{(k)}$.*

Remark 10.14.2 *Apart from the other conditions given in Theorem 10.14.1, those algorithms which fail to be globally convergent are most often because the map A is not closed. This happens because for these algorithms the direction finding map D is not closed. Thus Frank and Wolfe's method is globally convergent but Rosen's gradient projection method is NOT globally convergent. The main reason being that the direction finding map D for this method suddenly takes the steepest descent direction for the case when A_q has only one row and that too is required to be deleted.*

Remark 10.14.3 *Zangwill [173], Luenberger [106] and Bazaraa and Shetty [11] are excellent texts for understanding the global convergence theorem. These texts also prove the global convergence of some of the algorithms for UMP's and NLP's discussed here.*

10.15 Multistage Decision Problems and Dynamic Programming

We now present another powerful optimization technique developed by R. E. Bellman in 1957. This technique is known as *dynamic programming*, and is most suitable for solving those optimization problems which can be visualized as *multistage decision problems*. We shall restrict our discussion to deterministic finite stage decision problems whose basic characteristics are as follows

(i) There are finite number of *stages* $(i = 1, \ldots, N)$ in the problem.

(ii) Each stage has a number of *states* associated with it.

(iii) At each stage i, we have to take a *decision* $d(i)$ from amongst a number of possible choices, which may be infinite. The decision $d(i)$ depends upon the current state S_i of the system.

(iv) The effect of taking a decision $d(i)$ at stage i is to *change* the state of the system so that at stage $(i+1)$, the system is at a new state corresponding to the $(i+1)^{th}$ stage.

(v) At stage i of the system, a *return function* $R(i)$ is prescribed which depends upon the decision $d(i)$.

(vi) The sequence of decisions $d(i)$ taken at every stage, i.e. $\{d(1), \ldots, d(N)\}$, is called a *policy*.

(vii) There is an *overall return function* R which is a function of $(R(1), \ldots, R(N))$. Our objective is to find that policy which maximizes/minimizes the overall return function R. Such a policy, if it exists, is called an *optimal policy*.

The technique of dynamic programming is best suited when $R = \sum_{i=1}^{N} R(i)$ or

$$R = \prod_{i=1}^{N} R(i).$$

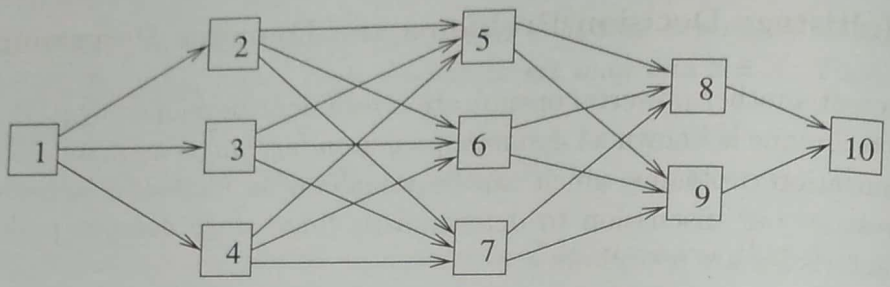
An optimization problem which can be visualized in the above framework can, in principle, always be solved by dynamic programming. Also, theoretically at least, the objective and constraint functions are not required to be convex or differentiable for the applicability of dynamic programming.

An Illustrative Example

We now take an illustrative example to understand the basic principle and the methodology of dynamic programming. The example taken here is a representative of the general class of problems commonly known as *network routing type problems*.

Let us consider the network consisting of 10 nodes and arcs as shown in Fig 10.6. The distances between those nodes which are connected by an arc are shown in the accompanying tables.

We are interested in finding the path of shortest distance between node 1 and node 10.



	2	3	4
1	2	4	3

	5	6	7
2	7	4	6
3	3	2	4
4	4	1	5

	8	9
5	1	4
6	6	3
7	3	3

	10
8	3
9	4

Fig. 10.6.

If we attempt to find the shortest path by trial and error or by complete enumeration, we immediately realize that none of these are feasible if the number of nodes are too large. However, if we are able to visualize the given problem as a multistage decision problem, then we can think of using dynamic programming to find the shortest path between node 1 and node 10.

Looking at the network, we observe that there are four stages at which the decision is to be taken. Initially we are occupying node 1 and that is the state of the system at the first stage. At this stage, a decision is to be taken whether to go to node 2, node 3 or node 4. The corresponding returns (as given in the first table) are 2, 4 and 3 units of distances, respectively. Once we reach the second stage, whose state is described by node 2, node 3 and node 4; a decision is to be taken whether to go to node 5, node 6 or node 7. The returns corresponding to this decision are given in the second table. If we continue in this manner then we can conclude that the stated problem is a four stage decision problem having overall return function $R = \sum_{i=1}^{4} R(i)$.

Once we are able to visualize the given optimization problem as a multistage decision problem and we decide to use the dynamic programming methodology, we have to make use of the most basic principle, namely, *Bellman's optimality principle*. This principle states that

" What ever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

There are couple of points which we will like to note here. There is nothing special about the word 'initial state' or 'initial decision'- the principle holds for any state and

any decision taken there. Also this principle probably reasserts the most natural and rational way of decision making.

Once we apply the Bellman's optimality principle to the given multistage decision problem, we shall get a recurrence relation; also called *Bellman's equation*. Further, there will be an initial (or terminal) condition from the system itself. Mathematically, then we simply need to solve the resulting Bellman's equation under the given initial/terminal condition.

Let us now go back to our illustrative example and recall that our aim is to find the path of shortest distance between node 1 and node 10. To make use of the optimality principle, let us define

$$f_i = \text{shortest distance from node } i \text{ to node } 10 \quad (i = 1, \ldots, 10).$$

Obviously, $f_{10} = 0$ and our aim is to find the value of f_1.

Now let us imagine that the current state of the system is node i (i.e. we are currently at node i corresponding to some intermediate stage). If we decide to go to any admissible node j (i.e. any node j which is joined by an arc with node i) and cover a distance of d_{ij}, then the optimality principle asserts that from node j to node 10 we should go via the path of shortest distance covering the distance of f_j units. Thus the total distance of this action is $(d_{ij} + f_j)$. Since the choice of j is arbitrary, if we find $\underset{j}{Min}\,(d_{ij} + f_j)$, it has to be the shortest distance from node i to node 10. Therefore, we get

$$f_i = \underset{\{j \text{ admissible}\}}{Min} (d_{ij} + f_j) \tag{10.32}$$

$$f_{10} = 0. \tag{10.33}$$

Since $f_{10} = 0$, using the recurrence relation (10.34), we can obtain f_9 and then $f_8, f_7, \ldots$ etc., getting f_1 finally. Thus we have

$$f_9 = Min\,(d_{9,10} + f_{10}) = Min\,(4 + 0) = 4$$

$$f_8 = Min\,(d_{8,10} + f_{10}) = Min\,(3 + 0) = 3$$

$$\begin{aligned} f_7 &= Min\,(d_{7,8} + f_8,\ d_{7,9} + f_9) \\ &= Min\,(3 + 4,\ 3 + 3) = 6 \end{aligned}$$

$$\begin{aligned} f_6 &= Min\,(d_{6,8} + f_8,\ d_{6,9} + f_9) \\ &= Min\,(6 + 3,\ 3 + 4) = 7 \end{aligned}$$

$$\begin{aligned} f_5 &= Min\,(d_{5,8} + f_8,\ d_{5,9} + f_9) \\ &= Min\,(1 + 3,\ 4 + 4) = 4 \end{aligned}$$

$$\begin{aligned} f_4 &= Min\,(d_{4,5} + f_5,\ d_{4,6} + f_6,\ d_{4,7} + f_7) \\ &= Min\,(4 + 4,\ 1 + 7,\ 5 + 6) = 8 \end{aligned}$$

$$f_3 = Min\,(d_{3,5} + f_5,\ d_{3,6} + f_6,\ d_{3,7} + f_7)$$
$$= Min\,(3+4,\ 2+7,\ 4+6) = 7$$

$$f_2 = Min\,(d_{2,5} + f_5,\ d_{2,6} + f_6,\ d_{2,7} + f_7)$$
$$= Min\,(7+4,\ 4+7,\ 6+6) = 11$$

$$f_1 = Min\,(d_{1,2} + f_2,\ d_{1,3} + f_3,\ d_{1,4} + f_4)$$
$$= Min\,(2+11,\ 4+7,\ 3+8) = 11.$$

Therefore the shortest distance between node 1 and node 10 is 11 units of distance. To obtain the actual path, we note that the minimum value 11 in f_1 is attained for node 3 and node 4. Therefore, from node 1 we can go to node 3 or node 4. To be specific, suppose we go to node 3. But then the optimality principle tells that from node 3 to node 10 we should go via the path of shortest distance, i.e. we should look for the value of f_3. There the minimum is attained for node 5, and so from node 3 we should go to node 5. Continuing in this manner, we get an optimal path as

$$1 \longrightarrow 3 \longrightarrow 5 \longrightarrow 8 \longrightarrow 10.$$

We can also check that the following paths are also optimal

$$1 \longrightarrow 4 \longrightarrow 5 \longrightarrow 8 \longrightarrow 10,$$

and

$$1 \longrightarrow 4 \longrightarrow 6 \longrightarrow 9 \longrightarrow 10.$$

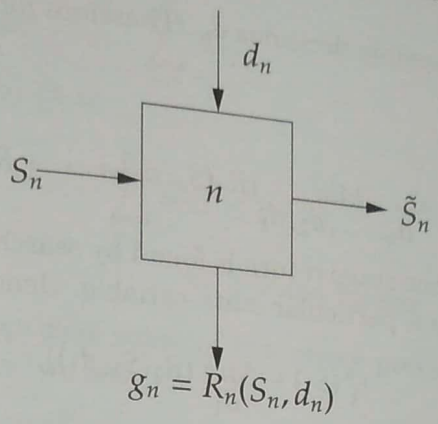
These paths are obtained if we take the other alternative optimal solution at node 1, and decide to go to node 4.

Remark 10.15.1 *We note that the optimal decision at every stage need not give an optimal policy. For example, the path obtained by taking optimal decision at every stage is,* $1 \longrightarrow 2 \longrightarrow 6 \longrightarrow 9 \longrightarrow 10$, *which has a total of 13 units of distance and therefore can not be optimal.*

Remark 10.15.2 *By defining* f_i *as the shortest distance from node 1 to node i, we get another recurrence relation with the initial condition as* $f_1 = 0$. *We then aim to find* f_{10}, *which can be obtained by evaluating* $f_2, f_3, \ldots$ *etc. recursively. Thus there are two possible approaches of employing dynamic programming methodology, namely, the forward recursion and the backward recursion. In practise, backward recursion is more popular because of its convenience and easy implementation.*

(Symbolic) Description of Dynamic Programming Model

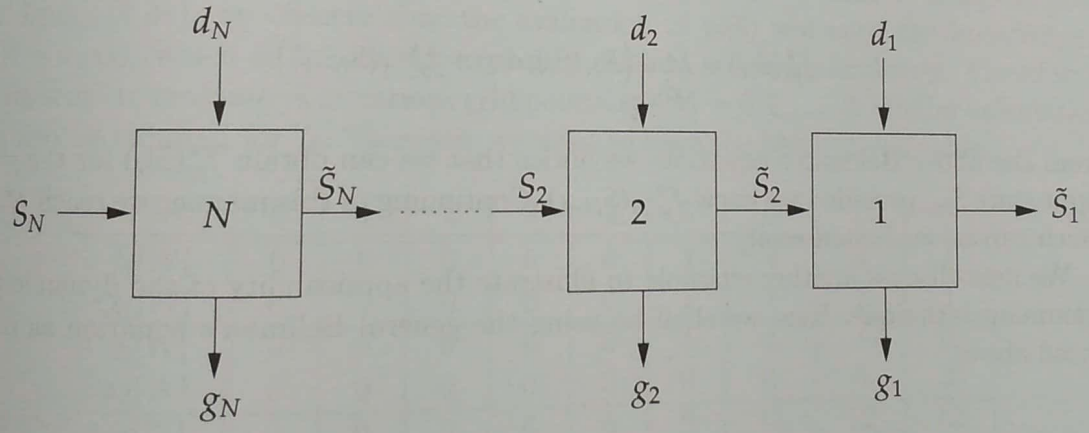
Let us denote the point at which we make a decision as a *stage*, and our input parameter as the *state*. Then for a single stage problem, we have



Here

n = stage number
S_n = input state
d_n = decision
$\tilde{S}_n$ = output state
g_n = return function $= R_n(S_n, d_n)$.

For an N-stage decision problem, we have



with $\tilde{S}_{i+1} = S_i \quad (i = 1, \ldots, (N-1)), \tilde{S}_2 = S_1$.

In dynamic programming, we solve multivariable optimization problems *sequentially*, or *one stage* at a time. Let $f_n(S_n, d_n)$ denote the accumulated total return calculated over n stages, given a particular state S_n. Further, let $f_n^\star(S_n)$ denote the optimal n-stage total return for a particular input state S_n. As a particular value of S_n might give rise to many possible decisions d_n, the optimal n-stage total return $f_n^\star(S_n)$ is attained for a

decision $d_n^\star$ amongst all possible decisions d_n. Therefore for the minimization case with $R = \sum_{i=1}^{N} R_i$, we have

$$f_n^\star(S_n) = \underset{d_n,\ldots,d_2,d_1}{Min} (R_n(S_n, d_n) + \cdots + R_1(S_1, d_1)).$$

Now the optimization one stage return is found by searching over all possible decision variables corresponding to a particular state variable. Hence,

$$f_1^\star(S_1) = \underset{d_1}{Min} (R_1(S_1, d_1)).$$

Here we note that the range of d_1 is determined by S_1. But S_1 is determined by what has happened in the previous stage which in turn gives

$$f_2^\star(S_2) = \underset{d_2}{Min} \Big(R_2(S_2, d_2) + f_1^\star(S_1)\Big).$$

Therefore for a general N-stage problem, with $R = \sum_{i=1}^{N} R_i$, we have

$$f_N^\star(S_N) = \underset{d_N}{Min} \Big(R_N(S_N, d_N) + f_{N-1}^\star(S_{N-1})\Big).$$

From the above Bellman's equation, we notice that we can obtain $f_N^\star(S_N)$ for the given input state S_N, provided we knew $f_{N-1}^\star(S_{N-1})$. Continuing in this manner we reach $f_1^\star(S_1)$ which can be evaluated easily.

We next discuss another example to illustrate the applicability of the dynamic programming technique. Here we shall be using the general Bellman's equation as introduced above.

Cargo Loading Problem

This problem has already been described in Chapter 1. Let there be N items. Also, let the weight of one unit of the ith item be w_i and the value of one unit of the ith item be v_i $(i = 1,\ldots,N)$. Further, let the maximum capacity of the cargo be N tons. The problem is to find the (integral) units of each item to be loaded so that the total value of the cargo is maximum. This leads to the following optimization problem

$$\text{Max} \quad \sum_{i=1}^{N} v_i x_i$$

subject to

$$\sum_{i=1}^{N} w_i x_i \leq W$$

$$x_i \geq 0 \text{ and integer} \quad (i = 1, \ldots, N).$$

Let $f_N(W)$ denote the optimal value of the above problem. Then, in view of our discussion as outlined earlier, we have

$$f_1(W) = \underset{x_1 = 0, 1, \ldots, \left\lfloor \frac{W}{w_1} \right\rfloor}{Max} (v_1 x_1)$$

and

$$f_N(W) = \underset{x_N = 0, 1, \ldots, \left\lfloor \frac{W}{w_N} \right\rfloor}{Max} \left(v_N x_N + f_{N-1}(W - w_N x_N) \right). \tag{10.34}$$

We now give actual calculations for $W = 8$, $N = 3$, $v_1 = 4$, $v_2 = 10$, $v_3 = 6$, $w_1 = 5$, $w_2 = 8$ and $w_3 = 3$. In terms of our notation of $f_N(W)$, we need to compute $f_3(8)$. But from (10.34), we observe that the evaluation of $f_3(8)$ will need the knowledge of $f_2(W - w_3 x_3)$, which is not known explicitly until x_3 is known explicitly. Therefore it makes sense to evaluate f_2 at various grid points, say, $W = 0, 1, \ldots, 8$. Similar calculations will also be required for f_1. Therefore we need to have the below given table

w →	0	1	2	3	4	5	6	7	8
$f_1[W]$	0	0	0	0	0	4	4	4	4
$x_1[W]$	0	0	0	0	0	1	1	1	1
$f_2[W]$	0	0	0	0	0	4	4	4	10
$x_2[W]$	0	0	0	0	0	0	0	0	1
$f_3[W]$	0	0	0	6	6	6	12	12	12
$x_3[W]$	0	0	0	1	1	1	2	2	2

Here $x_1[W]$ is the value of x_1 for which the maximum value $f_1[W]$ is attained. As an illustration, let us compute $f_1(5)$, $f_2(6)$ and $f_3(8)$. By definition

$$f_1(W) = \underset{x_1 = 0, 1, \ldots, \left\lfloor \frac{W}{w_1} \right\rfloor}{Max} (v_1 x_1).$$

As $w_1 = 5$, for $W = 5$, x_1 can take only values as $x_1 = 0$ and $x_1 = 1$. Therefore $f_1(5) = \max((4 \times 0), (4 \times 1)) = 4$, which is attained for $x_1 = 1$. Again, as

$$f_2(W) = \underset{x_2 = 0, 1, \ldots, \left\lceil \frac{W}{w_2} \right\rceil}{Max} \Big(v_2 x_2 + f_1(W - w_2 x_2) \Big),$$

and $w_2 = 8$; for $W = 6$, x_2 can take only value as $x_2 = 0$. Therefore $f_2(6) = ((0 \times 10) + f_1(8 - (8 \times 0)) = f_1(8) = 4$, which is attained for $x_2 = 0$. In a similar manner

$$f_3(W) = \underset{x_2 = 0, 1, \ldots, \left\lceil \frac{W}{w_3} \right\rceil}{Max} \Big(v_3 x_3 + f_2(W - w_3 x_3) \Big).$$

Since $w_3 = 3$, for $W = 8$, the possible values of x_3 are 0, 1, and 2. Therefore

$$\begin{aligned} f_3(8) &= \max \Big((6 \times 0) + f_2(8),\ (6 \times 1) + f_2(5),\ (6 \times 2) + f_2(2) \Big) \\ &= \max(10,\ 6 + 4,\ 12 + 0) = 12. \end{aligned}$$

Therefore the optimal value of the cargo is $f_3(8)$ which is equal to 12. Also the optimal value is attained for $x_3 = 2$. Now to determine x_2, we note that after loading 2 units of 3rd item, we have already consumed 6 units of weight. So, now the remaining capacity, namely $8 - 6 = 2$ tons must be loaded by remaining two items in an optimal manner. Hence we should look the value of $f_2(2)$ which is equal to zero and is attained for $x_2 = 0$. Next we need to check $f_1(2)$ which gives $x_1 = 0$. Therefore the optimal solution is $(\bar{x}_1 = 0,\ \bar{x}_2 = 0,\ \bar{x}_3 = 2)$ and the optimal value is 12.

The 'Curse of Dimensionality' in Dynamic Programming

In the cargo loading problem, the weight constraint $\sum_{i=1}^{N} w_i x_i \leq W$, gives rise to one state variable, namely the weight capacity of the cargo. Suppose we also have a volume constraint of the form $\sum_{i=1}^{N} q_i x_i \leq Q$, where q_i is the volume per unit of the ith item and Q is the total volume capacity of the cargo. Then along with W, we have another state variable Q and we need to define $f_N(W, Q)$ similar to $f_N(W)$. The solution of the problem will require either total exhaustive search (for the continuous case) or grid search (for the discrete case) over the domain of (W, Q). Thus each constraint linking the decision variables in functional form requires the addition of one state variable. Further, in our computations, either all values of the states or values at several grid points are examined. Therefore presence of several state variables increases greatly the total amount of computations needed to solve a given problem.

In dynamic programming terminology, an allocation problem with one constraint is designated as a *one dimensional problem*, even though it may contain several decision

variables. So, here the number of constraints determines the dimension of the problem, and each constraint gives rise to a state variable which may take several possible values. Therefore if the number of constraints is large, we have large number of state variables (i.e. the problem has large dimension) and therefore large number of function evaluations and related optimization. Since, in order to compute an optimal solution of the given problem, we need to retain an optimal solution to every state variable combination at each stage, for large dimensional problems the amount of information required (both to be computed and stored) would be astronomical! This dramatic increase in the amount of computation and storage required to solve a given optimization problem has been called the *curse of dimensionality* by Bellman [14]. Bellman and Dreyfus [15] have suggested a procedure that can be used to reduce the *curse of dimensionality* by the introduction of Lagrange multipliers into the problem. Some more recent methods, e.g. 'approximate dynamic programming' are presented in Bertsekas [19].

10.16 Summary and Additional Notes

- Though there are many algorithms for solving NLP's, we have discussed only some chosen few in our presentation here. For linearly constrained NLP's, the algorithms discussed are Frank and Wolfe's method and Rosen's gradient projection method which are presented in Sections 10.2 and 10.4 respectively.
- For solving general NLP's, which may have nonlinear constraints, we have concentrated only on the penalty function method and the barrier function method. These methods solve a given NLP by solving a sequence of unconstrained optimization problems, and therefore appropriately called sequential unconstrained minimization techniques (SUMT).
- Theoretically, SUMT is capable of solving even nonconvex NLP's provided we have methods to find the global min point of resulting unconstrained minimization problems. Though, in general, it is not easy to obtain the unconstrained min point of a nonconvex function, we shall have opportunity to discuss some appropriate evolutionary methods for the same in the later part of the book.
- Section 10.5 discuss the motivation for the penalty function method while Section 10.6 and 10.7 respectively present the analytical and numerical implementation of the same. The mathematical justification of the penalty function method is detailed in Section 10.8.
- Section 10.9 gives a motivation for the barrier function method while its analytical and numerical implementations are presented in Sections 10.10 and 10.11 respectively.
- Though the penalty function approach and the barrier function approach have been studied at different times with different names (e.g. Caroll [32] studied the barrier function approach under the name *created response surface technique*), it was Fiacco and McCormick who thoroughly investigated and popularized these methods

for solving practical problems in a series of papers in mid and late sixties. Their book entitled 'Nonlinear Programming: SUMT [57]' is now a classic on this subject. Fiacco and McCormick in 1968 also introduced the mixed penalty-barrier auxiliary function approach for solving NLP's where both, equality and inequality constraints are present.

- Fletcher [58] in 1970 developed an *exact penalty function method* for solving NLP's where a single unconstrained minimization yields a solution to the original problem.
- Other useful methods for solving NLP's (which have not been discussed here) are Zoutendijk's feasible direction method, Wolfe's reduced gradient method, Zangwill's convex-simplex method. Some appropriate references for these methods are the texts by Avriel [6], Simmons [143], Bazaraa and Shetty [11], Luenberger [106], Zangwill [173] and Kambo [89].
- One of the major demerit of penalty function method is that with large values of the penalty parameter α, the Hessian $\frac{\partial^2 q}{\partial x^2}$ of the penalized function $q(x, \alpha)$ becomes ill-conditioned, (i.e. the ratio of the maximum eigenvalue to the minimum eigenvalue of the matrix is fairly large), leading to large computational errors. To reduce the problem of ill conditioning in the penalty function method, M. R. Hestenes in 1969 and also M. J. D. Powell in 1969, proposed the *augmented Lagrangian method* to solve equality constrained NLP's. Later, in 1973, R. T. Rockafellar gave an extension of this method to the inequality constraints. For the following constrained optimization problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ &&& g_i(x) \le 0, \quad (i = 1, \ldots, m), \\ &&& h_j(x) = 0, \quad (j = 1, \ldots p), \end{aligned}$$

the augmented Lagrangian function is defined as

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) + \frac{1}{2}\alpha \Big(\sum_{i=1}^{m} \max\{0, g_j(x)\}^2 + \sum_{j=1}^{p} h_j^2(x) \Big),$$

which differs from the usual Lagrangian by the presence of squared penalty term. The penalty parameter $\alpha > 0$ is taken just large enough to make the Lagrangian function convex in x in a neighborhood of an exact optimal solution x^*, and not indefinitely large to safeguard against ill-conditioning. The local convexity of the Lagrangian function ensures the applicability of the duality results and consequently equality of the primal-dual optimal objective values. During the course of the augmented Lagrangian penalty algorithm, the Lagrange multipliers (λ, μ) are updated so as to approach the dual optimal values (λ^*, μ^*) such that $L(\cdot, \lambda^*, \mu^*)$ remains convex even with finite value of α. A good discussion of this method is given in Bertsekas [20], Fletcher [58] and Nocedal and Wright [119].

- Another effective class of methods for solving nonlinearly constrained optimization problems are based on *sequential quadratic programming (SQP) approach*. The SQP methods can be implemented both in line search and trust region frameworks, and are very effective for solving NLP's having significant nonlinearities. A good discussion on SQP methods is given in Nocedal and Wright[119].
- The most basic books on dynamic programming are Bellman [14] and, Bellman and Dreyfus [15]. The book by Bertsekas [19] presents a very detailed application of the dynamic programming technique to problems of optimal control.

10.17 Exercises

10.1 *Starting with* $(x_1 = 1, x_2 = 4)$, *solve the following problem by Frank and Wolfe's method*

$$\begin{aligned} &\textit{Min} \quad (x_1 - 4)^2 + x_2^2 \\ &\textit{subject to} \\ &\qquad x_1 + 2x_2 \leq 9 \\ &\qquad -x_1 + x_2 \leq 3 \\ &\qquad x_1, x_2 \geq 0. \end{aligned}$$

10.2 *Determine the projection of the gradient of the function* $f(x) = 5x_1 - 3x_2 + 6x_3$ *onto the* x_1x_2*-plane. Sketch both, the gradient and its projection.*

10.3 *In solving the nonlinear programming problem*

$$\begin{aligned} &\textit{Max} \quad 2(x_1 - 1)^2 - x_2^2 + 16x_3 \\ &\textit{subject to} \\ &\qquad x_1 + x_2 \leq 8 \\ &\qquad x_1 + 3x_3 \leq 9 \\ &\qquad x_2 - 2x_3 \leq 0 \\ &\qquad x_1, x_2, x_3 \geq 0\,, \end{aligned}$$

what direction of movement from each of the following points would Rosen's gradient projection method select?

(*i*) $(3,4,2)$ (*ii*) $(0,0,3)$ (*iii*) $(6,2,1)$ (*iv*) $(1,4,2)$.

10.4 *Solve the following problem via Rosen's gradient projection method, using origin as the starting point*

$$\begin{aligned} \textit{Max} \quad & 2x_1^2 + x_2 + x_1^3 x_2 \\ \textit{subject to} \quad & \\ & 2x_1 + x_2 \leq 8 \\ & x_1 - 2x_2 \leq 4 \\ & -x_1 + 2x_2 \leq 0 \\ & x_1, x_2 \geq 0. \end{aligned}$$

10.5 *Consider the constraints* $Ax \leq b$ *and let* $P = I - A_1^T(A_1A_1^T)^{-1}A_1$, *where* A_1 *represents the gradient of the binding constraints at a given feasible point* $\hat{x}$. *What are the implications and geometric interpretations of following statements?*

(i) $P(\nabla f(\hat{x}))=0$

(ii) $P(\nabla f(\hat{x})) = \nabla(f(\hat{x}))$

(iii) $P(\nabla f(\hat{x})) \neq 0$.

10.6 *Using the analytical implementations of the penalty function method, solve the following problem*

$$\begin{aligned} \textit{Max} \quad & 4x - x^2 \\ \textit{subject to} \quad & \\ & x \leq 1. \end{aligned}$$

10.7 *Using the analytical implementation of the barrier function method, solve the following problem*

$$\begin{aligned} \textit{Max} \quad & x_1 \\ \textit{subject to} \quad & \\ & x_1 \leq x_2 \\ & x_2 \leq 2. \end{aligned}$$

10.8 *Using the numerical implementation of the barrier function method, solve the following*

$$\begin{aligned} \textit{Max} \quad & (x_1 - 5)^2 + (x_2 - 3)^2 \\ \textit{subject to} \quad & \\ & x_1 + x_2 \leq 3 \\ & -x_1 + 2x_2 \leq 4, \end{aligned}$$

by taking the starting point as $(0,0)^T$. *Can the barrier function method be started from the point (2,1)?*

10.9 *Consider the problem*

$$\begin{aligned} &\textit{Max} && x_1^2 - 2x_1 - x_2 \\ &\textit{subject to} \\ & && x_1 + 5x_2 \le 12 \\ & && x_1 + x_2 \ge 2 \\ & && x_1, x_2 \ge 0, \end{aligned}$$

and solve the same by (i) Frank and Wolfe's method (ii) Rosen's gradient gradient projection method (iii) the penalty function method and (iv) the barrier function method.

10.10 *Show that for the nonlinear programming problem 'min $f(x)$, subject $g_i(x) \le 0\,(i = 1, 2, \ldots, m)$', the function $B_L(x)$ given by*

$$B_L(x) = \sum_{i=1}^{m} \log_e(-g_i(x)),$$

meets the requirements of a barrier function. The function $B_L(x)$ is called the logarithmic barrier function (Here, obviously, the domain of B_L is the interior of the feasible region of the given NLP.)

Use the logarithm barrier function method to solve the problem

$$\begin{aligned} &\textit{Min} && x \\ &\textit{subject to} \\ & && 0 \le x \le 1. \end{aligned}$$

10.11 *A student has to be take examination in three courses X, Y, and Z. He has three days available for study. He feels that it will be best to devote a whole day to the study of the same course, so that he may study a course for one day, 2 days, 3 days or not at all. His estimate of grades that he may get by the study are*

study day \ course	X	Y	Z
0	1	2	1
1	2	2	2
2	2	4	4
3	4	5	4

How should he plan to study so as to maximize his sum of grades.

10.12 *Maximize hydroelectric power $p(x)$, $X = (x_1, x_2, x_3)$, produced by building dams on three different river basins, where*

$$p(x) = f_1(x_1) + f_2(x_2) + f_3(x_3)$$

Here $f_i(x_i)$ is the power generated from the ith basin by investing rupees x_i (in Lakhs). The total budgetery provision is $x_1 + x_2 + x_3 \leq 3$ (in Lakhs). The functions f_1, f_2, and f_3 are given in the following table and integer solution of the problem is required

$f_j(x_i)$ \ x_i	0	1	2	3
f_1	0	2	4	6
f_2	0	1	5	6
f_3	0	3	5	6

11 Computational Complexity and Karmarkar's Algorithm for Linear Programming

11.1 Introduction

We have already studied the simplex algorithm for solving linear programming problems. This algorithm has been applied to numerous real life problems of different sizes and many efficient codes are available for its easy implementation.

The basic aim of this chapter is to discuss certain *complexity* issues related to the simplex algorithm and thereby conclude that the *worst case computational complexity* of the simplex algorithm is *exponential*. In this sense the simplex algorithm is not a *good* algorithm because theoretically an algorithm is considered to be a good algorithm if its worst case computational complexity is *polynomial*. Roughly speaking if the worst case computational complexity of an algorithm is exponential, then the *amount of computation* will grow exponentially with respect to the problem *size*, thereby making some problems of very large size almost impossible to be solved. This observation raises the question: *Does there exist a polynomial time algorithm for solving LPP's* ?, i.e. does there exist an algorithm for solving LPP's where the amount of computation grows like a polynomial, with respect to the problem size? This question has been answered in the affirmative by N. Karmarkar in 1984 when he proposed the *projective scaling algorithm* for solving LPP's.

In the literature these are many variants of the original Karmarkar's algorithm but in our presentation here, we shall mostly stick to the basic algorithm as developed by Karmarkar. Also at places, we may not be very precise mathematically but every effort will be made to understand and clarify various issues involved in the development of Karmarkar's algorithm.

One important aspect of Karmarkar's algorithm is the fact that it is an *interior point method*(IPM). Such algorithm have become very important recently because of their applications in *semidefinite programming* and *second order cone programming*.

11.2 Simplex Algorithm: A Review of the Basic Facts

The most basic characteristic of a linear programming problem is the fact that its objective and constraint functions are linear functions of decision variables. This structure of linearity guarantees the following for LPP's

(P1) The feasible region of a LPP is always a convex set.

(P2) If the given LPP has an optimal solution then at least one corner point of the feasible region is optimal.

(P3) Every local optimal solution of the given LPP is also a global optimal solution.

Because of the property (P2), it makes sense to concentrate only on the corner points of the feasible region, i.e. given the LPP, we determine all corner points of the feasible region and then choose the one at which the objective function value is optimum.

Since having all corner points at one go may be difficult, Dantzig proposed the following iterative procedure called the *simplex algorithm*

Step 1 Start from an initial corner point.

Step 2 Check if the given (current) point is optimal? If yes, stop; otherwise go to Step 3. (Here it may be noted that this step essentially involves comparing the current objective function value vis-a-vis values at those corner points which are joined by an edge with the current corner point).

Step 3 From the current corner point, move to an adjacent corner point (that is, a corner point which is joined by an edge with the current corner point) so that the objective function value is improved, and then go to Step 2.

Thus the simplex algorithm moves from a given corner point of the feasible region to another corner point which is joined by an edge and the objective function value is improved. Since the number of corner points is finite the simplex algorithm is a *finite iterative procedure*(except in certain rare situations of *cycling*) which will always give an optimal solution to the given LPP, if there is one. It may be interesting to know *How big this finite number is*, and *How does it grow with the size of the LPP?*

By reading Steps 1-3, one may wonder how the simplex algorithm will be implemented if the number of variables are more than 2 (or 3) because the algorithm has been stated in terms of 'corner points' and 'corner point' is a geometric concept. This difficulty is resolved by defining the 'corner point' algebraically in terms of the *basic feasible solution* and replacing the word 'corner point' in Steps 1-3 by the word 'basic feasible solution'. Since basic feasible solution is an equivalent algebraic analogue of the 'corner point', it can be computed (at least theoretically) no matter how many variables and constraints are present in the given LPP. The actual computational details of the simplex algorithm are not described here as they are already presented in Chapters 2 and 3. Also these details are not really relevant for our discussion here as long as we understand the simplex algorithm geometrically in terms of its movement from one corner point to another corner point along edges till an optimal corner point is obtained.

11.3 Computational Complexity of the Simplex Algorithm: Definitions

Consider the linear programming problem

$$\begin{aligned} &\text{Max} && z = c^T x \\ &\text{subject to} \\ & && Ax = b \\ & && x \geq 0, \end{aligned} \tag{11.1}$$

where $c \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ and $A = (a_{ij}) \in \mathbf{R}^{m \times n}$ a $(m \times n)$ matrix, with $b \geq 0$ and Rank $A = m (< n)$. We further assume that the data, namely the elements of c, b, and A are *rational*. In this case, we can multiply by a suitable integer and scale each variable appropriately so that all data is in integers only.

We wish to answer the question: *How much computational effort is involved in solving any instance of such a LPP?* Theoretically such questions about an algorithm are answered in terms of their *complexity*.

Definition 11.3.1 *Let $f(n)$ and $g(n)$ be functions from positive integers to positive reals. Then f is said to be of order $O(g(n))$, written as $f(n) = O(g(n))$, if there exists $\alpha > 0$ such that for large enough n, $f(n) \leq \alpha g(n)$.*

In complexity theory we are interested in the behavior of algorithm when supplied with very large *input*, that is, we wish to know how does the *amount of computation* grow with respect to *input size*. Thus our interest here is not in the *amount of computation* for *a problem* but rather in its growth as the inputs of very large size are supplied.

Using this notion, the complexity of an algorithm may be expressed as in phrases like: *the time complexity of the given algorithm is* $O(n^3)$. To have some idea about the growth of certain functions, suppose that for $n = 10$, there is an algorithm of order $O(\log n)$ that requires a full hour, an $O(n^5)$ algorithm that requires only one minute, and $O(2^n)$ algorithm that needs just one second. All these three algorithms are capable of solving any instance of the given problem. It can now be computed that for $n = 100$, the $O(\log n)$ algorithm will require 2 hours, the $O(n^5)$ algorithm will require 69.4 days and the $(O(2^n))$ will need 10^{17} centuries! As problem size becomes large, such dramatic differences clearly make $O(\log n)$ algorithm much preferred to $O(n^5)$ and $O(n^5)$ algorithm much preferred to $O(2^n)$ algorithm.

We measure the complexity of an algorithm as a function of the size of the input of the algorithm. To submit this input for solution by a computer, we have to encode it, that is, to represent it as a sequence of symbols over some fixed alphabet such as bits, for example we can have binary encoding. Thus input of an algorithm is represented as a sequence (or string) of symbols. We define the size of the input as the *length* of this string, that is, the number of symbols in it.

In the context of the linear programming problem (11.1), the data is m, n, A, c and b. To define the size of problem (11.1), we have to find the total length of the string when the integral data m, n, A, c and b is encoded in terms of binary numbers. For a real number v, let $\lceil v \rceil$ denote the *smallest* integer which is more than or equal to v. The integer $\lceil v \rceil$ is called the *ceiling* of the real number. We know that the number of digits in binary encoding of an integer p is $\lceil (1 + log_2(1 + |p|)) \rceil$. So when all the data of the given LPP is integral, its size is defined as

$$\text{size} = \lceil (1 + log_2(1 + m)) \rceil + \lceil (1 + log_2(1 + n)) \rceil + \sum_j (1 + \lceil log_2(1 + |c_j|) \rceil) +$$

$$\sum_i (1 + \lceil log_2(1 + |b_i|) \rceil) + \sum_i \sum_j (1 + \lceil log_2(1 + |a_{ij}|) \rceil) .$$

Since our interest is only in finding the upper bound on the computational effort involved, we need not compute the size of LPP (11.1) exactly. Infact some lower bound L on the actual number of bits required to record the data is enough provided we can show that there is a polynomial growth in the overall computational effort which is an increasing function of the number L. Therefore for the LPP (11.1) we take

$$L = \lceil (1 + log_2 m + log_2 n + \sum_j (1 + log_2(1 + |c_j|)) + \sum_i (1 + log_2(1 + |b_i|)) +$$

$$\sum_i \sum_j (1 + log_2(1 + |a_{ij}|)) \rceil$$

as the size of LPP (11.1) and not the actual size as defined above.

The main computational step in the simplex algorithm is the pivot step. Hence the effort involved in solving the given LPP by the simplex algorithm can be measured by the number of pivot steps made before the algorithm terminates. If a basic feasible solution of the LPP (11.1) is not known, then the simplex algorithm is applied twice, first on the Phase-I problem and then on the given problem itself, starting with the basic feasible solution provided by the Phase-I solution. So from a practical point of view the computational effort required to solve the LPP(11.1) for which a feasible basis is not known is approximately twice the effort required for solving a comparable problem in which the feasible basis is available for starting the simplex algorithm. Hence in the following we assume that a feasible basis is available. Then we wish to answer the question

'*Starting with a known feasible basis for the given LPP(11.1) (arbitrary), how many pivot steps are required before the simplex algorithm terminates?*'

Since we do not wish to answer this question for a particular given problem but rather for any arbitrary instance of the class of problems, a more meaningful question is as follows

'Given the size, what is a mathematical upper bound on the number of pivot steps required for solving any LPP(11.1) of that size by the simplex algorithm, starting with a known feasible basis?'

This upper bound determines, what is known as the *worst case computational complexity* of the simplex algorithm. Thus the worst case computational complexity provides a guaranteed upper bound on the computational effort needed to solve any instance of LPP(11.1) by the simplex algorithm as a function of its size.

Definition 11.3.2 (Polynomial Time Algorithm). *An algorithm for solving a problem is said to be a polynomial time algorithm (or polynomially bounded algorithm) if the worst case computational effort required for solving the problem, using, the given algorithm, is bounded above by a polynomial in the size of the problem.*

In other words, if L is the input size and the number of arithmetic operations is less than or equal to $\alpha f(L)$ for some $\alpha > 0$ and $f(L)$ a polynomial of fixed degree in L, then the algorithm is a polynomial time algorithm. Therefore for a polynomial time algorithm, the amount of computation is never greater than some fixed power of L, i.e. $aL^b, a > 0, b > 0$, no matter which problem is solved.

Now in context of the simplex algorithm, there are at most ${}^nC_m$ vertices that the algorithm could possibly visit. But

$$ {}^nC_m = \frac{n!}{m!(n-m)!} = \left(\frac{n}{m}\right)\left(\frac{n-1}{m-1}\right)\cdots\left(\frac{n-(m-1)}{m-(m-1)}\right) > \left(\frac{n}{m}\right)^m $$

which is at least 2^m whenever $n \geq 2m$. Therefore it makes sense to think that on some problems, the amount of computations needed could be of exponential order. This is indeed true as shown by Victor Klee and George Minty in 1971. We discuss these details in the next section.

11.4 Simplex Algorithm is not a Polynomial Time Algorithm

We shall here show that for every n we can construct a LPP having 2^n corner points such that the simplex algorithm is forced to visit each of these extreme points. This will prove that for every n, there is an instance of an LPP for which the simplex algorithm is forced to take exponential number of iterations and thus the worst case computational complexity of the algorithm will turn out to be exponential.

Definition 11.4.1 (Isotonic Path). *Let S be the set of feasible solutions of the given LPP. A sequence of points $\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(p)}\}$ is said to be an isotonic path if*

(i) each point in the sequence is a corner point(basic feasible solution)
(ii) consecutive points in the sequence are adjacent corner points i.e. the basis matrices

of the corresponding basic feasible solutions differ only by one column (iii) as we proceed the objective function value improves, i.e. for the case of minimization, $z(x^{(0)}) > z(x^{(1)}) > z(x^{(2)}) > \ldots > z(x^{(p)})$.

Given an isotonic path $\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(p)}\}$, the number of points in the sequence (excluding the initial point) is called the *length* of the isotonic path. Thus for the sequence $\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(p)}\}$ the *length of the isotonic path* is p. Note that if an isotonic path is found in S, the simplex algorithm can be made to follow this path by making appropriate dropping and entering vector selection.

Example 11.4.1 *For the linear programming problem*

$$\begin{aligned} \textit{Max} \quad & z = 4x_1 + 3x_2 \\ \textit{subject to} \quad & \\ & x_1 + x_2 \leq 8 \\ & 2x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{11.2}$$

Identify all isotonic paths of length 2. Is there an isotonic path of length 3 ?

Solution This problem has already been solved by the graphical method in Chapter 2. The feasible region has four corner points, namely, O, P, Q and R (see Fig 11.1) with objective function values as $z = 0, z = 20, z = 26$, and $z = 24$ respectively. Therefore if we denote these corner points as $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$, respectively then $\{x^{(0)}, x^{(1)}, x^{(2)}\}$ and $\{x^{(0)}, x^{(3)}, x^{(2)}\}$ are two isotonic paths of length 2. Also there is no isotonic path of length 3.

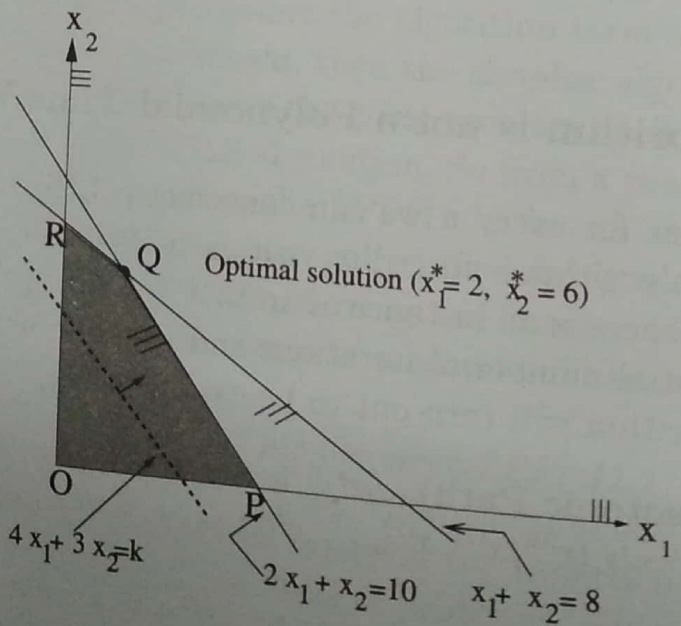


Fig. 11.1.

Here it may be noted that $\{x^{(0)}, x^{(2)}\}$ is not an isotonic path because these are not adjacent corner points, even though $z(x^{(2)}) = 26 > 0 = z(x^{(0)})$.

Theorem 11.4.1 *For every $n > 1$, there is an LPP with $2n$ equations, $3n$ variables and integer coefficients with absolute value bounded by 4, such that the simplex algorithm in the worst case may take $(2^n - 1)$ iterations to find the optimal solution.*

We shall not prove this theorem but rather give the construction of the desired LPP, called *Klee and Minty cube* in the literature. We shall observe that the theorem holds for $n = 2$ and $n = 3$, and thereby convince ourselves that it holds for any general n as well.

The Klee and Minty Cube

We shall first give the construction for $n = 2$ and $n = 3$, which will then lead to the construction of the desired LPP for general n.

Problem 1 (n=2)

$$\begin{aligned} \text{Min} \quad & -x_2 \\ \text{subject to} \quad & \\ & \epsilon \le x_1 \le 1 \\ & \epsilon x_1 \le x_2 \le 1 - \epsilon x_1 \\ & x_1, x_2 \ge 0. \end{aligned} \tag{11.3}$$

where $0 < \epsilon < \frac{1}{2}$.

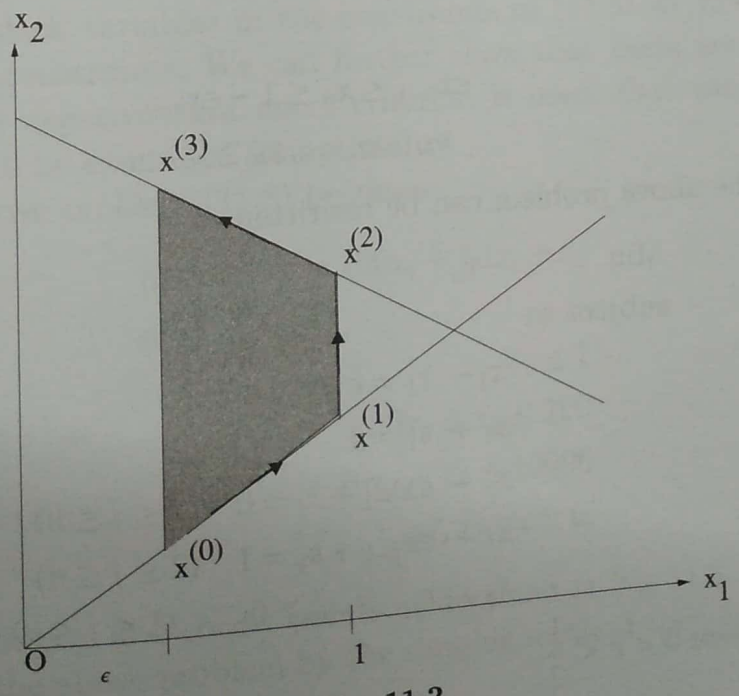


Fig. 11.2.

The corner points (basic feasible solutions) for this problem are (see Fig 11.2) $x^{(0)} = (\epsilon, \epsilon^2)$, $x^{(1)} = (1, \epsilon)$,, $x^{(2)} = (1, 1-\epsilon)$,, $x^{(3)} = (\epsilon, 1-\epsilon^2)$, and $\{x^{(0)}, x^{(1)}, x^{(2)}, x^{(3)}\}$ constitutes an isotonic path of length $(2^2 - 1) = 3$. So there are *four* corner points and in the worst case the simplex algorithm may be forced to visit each of these corner points.

Problem 2 (n=3)

$$\begin{aligned} \text{Min} \quad & -x_3 \\ \text{subject to} \quad & \\ & \epsilon \leq x_1 \leq 1 \\ & \epsilon x_1 \leq x_2 \leq 1 - \epsilon x_1 \\ & \epsilon x_2 \leq x_3 \leq 1 - \epsilon x_2 \\ & x_1,\ x_2, x_3 \geq 0, \end{aligned} \tag{11.4}$$

where $0 < \epsilon < \frac{1}{2}$.

The set of feasible solutions for this problem is slightly perturbed cube in $\mathbf{R}^3$. This problem has eight basic feasible solutions given by $x^{(0)} = (\epsilon, \epsilon^2, \epsilon^3)$, $x^{(1)} = (1, \epsilon, \epsilon^2)$, $x^{(2)} = (1, 1-\epsilon, \epsilon-\epsilon^2)$, $x^{(3)} = (\epsilon, 1-\epsilon, \epsilon-\epsilon^3)$, $x^{(4)} = (\epsilon, 1-\epsilon, 1-\epsilon+\epsilon^3)$, $x^{(5)} = (1, 1-\epsilon, 1-\epsilon+\epsilon^2)$, $x^{(6)} = (1, \epsilon, 1-\epsilon^2)$, $x^{(7)} = (\epsilon, \epsilon^2, 1-\epsilon^2)$, and $\{x^{(0)}, x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}, x^{(6)}, x^{(7)}\}$ constitutes an isotonic path of length $(2^3 - 1) = 7$.

Problem 3 (general n)

$$\begin{aligned} \text{Min} \quad & -x_n \\ \text{subject to} \quad & \\ & \epsilon \leq x_1 \leq 1 \\ & \epsilon x_1 \leq x_2 \leq 1 - \epsilon x_1 \\ & \vdots \\ & \epsilon x_{n-1} \leq x_n \leq 1 - \epsilon x_{n-1} \\ & x_1,\ x_2, \ldots, x_n \geq 0, \end{aligned} \tag{11.5}$$

where $0 < \epsilon < \frac{1}{2}$. The above problem can be rewritten as

$$\begin{aligned} \text{Min} \quad & -x_n \\ \text{subject to} \quad & \\ & x_1 - r_1 = \epsilon \\ & x_1 + s_1 = 1 \\ & x_j - \epsilon x_{j-1} - r_j = 0 \quad (2 \leq j \leq n) \\ & x_j + \epsilon x_{j-1} + s_j = 1 \quad (2 \leq j \leq n) \\ & x_j,\ r_j,\ s_j,\ \geq 0 \quad (1 \leq j \leq n), \end{aligned} \tag{11.6}$$

where $0 < \epsilon < \dfrac{1}{2}$. (11.7)

This problem has $2n$ equations and $3n$ variables. Taking $\epsilon = \frac{1}{4}$, the coefficients in the constraint equations are integers and absolute value bounded by 4. Further it can be shown that there are 2^n basic feasible solutions (corner points) and there is an isotonic path of length $(2^n - 1)$. Therefore in the worst case the algorithm may be forced to visit each of these corner points and thus takes exponential number of iterations.

This example was constructed by Klee and Minty and the general problem is called the *Klee and Minty cube*.

Another problem, similar to the Klee and Minty cube, can be constructed which also demonstrates that in the worst case, the computational complexity of the simplex algorithm is exponential. Here the entry criterion of the simplex algorithm is taken as negative most value of $(z_j - c_j)$ rather than the first negative value of $(z_j - c_j)$ encountered (as in the Klee and Minty cube). We describe the problem in the following.

A Variant of the Klee and Minty Cube

We consider the LPP

$$\text{Max} \qquad z = \sum_{j=1}^{n} 10^{n-j} x_j$$

subject to

$$\left(2 \sum_{j=1}^{i-1} 10^{i-j} \; x_j\right) + x_i \leq 100^{i-1} \quad (1 \leq i \leq n)$$

$$x_j \geq 0 \quad (1 \leq j \leq n)\,. \qquad (11.8)$$

After adding n slack variables in the constraints of (11.8) we note that, there are $2n$ variables and n constraints. We can further show that there are 2^n basic feasible solutions and if the negative-most entry criterion is used, then each of the 2^n basic feasible solutions will be examined for optimality.

For $n = 3$ the above problem (11.8) becomes

$$\begin{aligned} \text{Max} \qquad & z = 100x_1 + 10x_2 + x_3 \\ \text{subject to} \qquad & \\ & x_1 \leq 1 \\ & 20x_1 + x_2 \leq 100 \\ & 200x_1 + 20x_2 + x_3 \leq 10000 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \qquad (11.9)$$

having the feasible region and corner points as depicted in Fig 11.3.

If we now solve the above problem by the simplex method then following will be the initial simplex tableau.

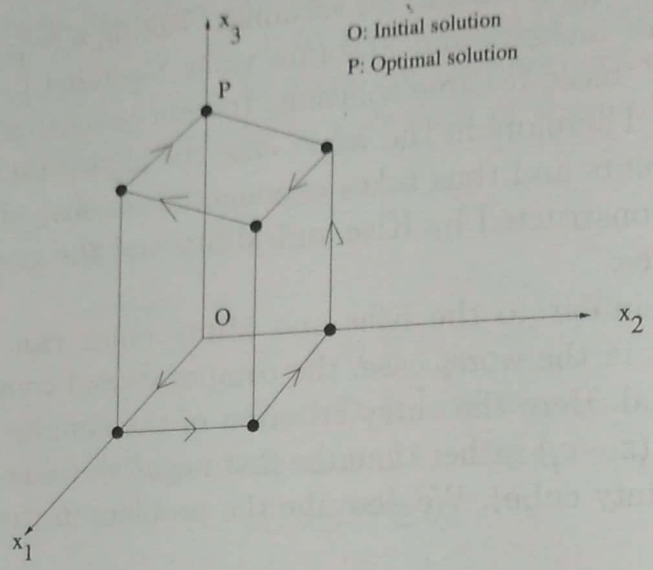


Fig. 11.3.

		$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(s_1)}$	$y^{(s_2)}$	$y^{(s_3)}$
←	$s_1 = 1$	1	0	0	1	0	0
	$s_2 = 100$	20	1	0	0	0	0
	$s_3 = 10000$	200	20	1	0	1	1
	0	−100	−10	−1	0	0	0
		↑					

The b.f.s of the subsequent iterations of the simplex algorithm are summarized in the following table

Iteration no.	b.f.s	z
1	$s_1 = 1\ s_2 = 100\ s_3 = 10000$	0
2	$x_1 = 1\ s_2 = 80\ \ s_3 = 9800$	100
3	$x_1 = 1\ x_2 = 80\ \ s_3 = 8200$	900
4	$s_1 = 1\ x_2 = 100\ s_3 = 8000$	1000
5	$s_1 = 1\ x_2 = 100\ x_3 = 8000$	9000
6	$x_1 = 1\ x_2 = 80\ \ x_3 = 8200$	9100
7	$x_1 = 1\ s_2 = 80\ \ x_3 = 9800$	99100
8	$s_1 = 1\ s_2 = 100\ x_3 = 10000$	10000

So there are *eight* basic feasible solutions numbered above and $\{x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(7)}\}$ constitute an isotonic path of length $2^3 - 1 = 7$. This path starts from the origin O and terminates at the point P visiting all corner points as depicted in Fig. 11.3.

11.5 Karmarkar's Algorithm for Linear Programming

In this section we present the basic ideas of the projective scaling algorithm due to Karmarkar [91], which is a polynomial time algorithm for solving LPP's. Here the emphasis is more on the geometrical understanding rather than on the detailed mathematical derivation.

The Basic Idea

As mentioned by Karmarkar, his approach is based on two fundamental insights, namely,

(i) If the current solution is near the centre of the polytope, it makes sense to move in the direction of steepest descent (when the objective function is to be minimized).
(ii) The solution space can be transformed so as to place the current point near the centre of the polytope in the transformed space without changing the problem in any essential way.

We can see the first point in Fig 11.4. below. As $x^{(0)}$ is near the centre of the polytope, one can improve the solution substantially by moving in the direction of steepest descent. But if $x^{(1)}$ is so moved, it will hit the boundary of the feasible region before much improvement occurs.

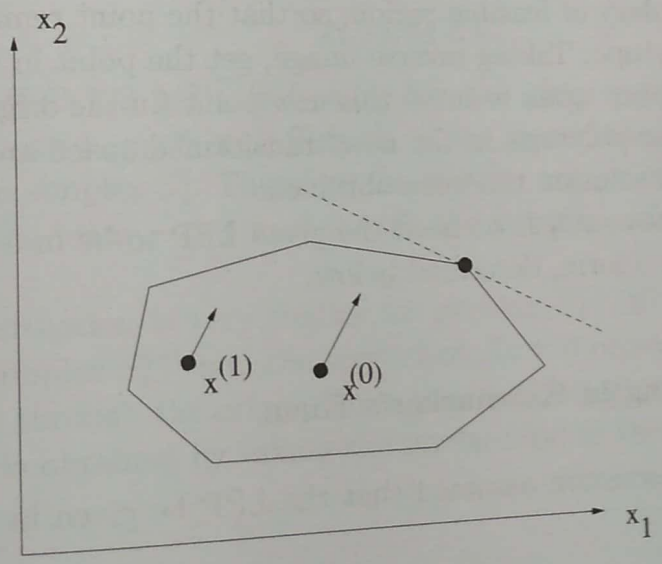


Fig. 11.4.

The second point is not so obvious. In fact it is very novel and original idea which makes Karmarkar's algorithm really unique. In this context we observe that when we write down the data defining the given LPP, we, in a sense overspecify the problem. The scaling of data is for instance quite arbitrary. We can switch from meter to centimeter without changing the problem in any important sense. Yet the transformation of data

that leaves the problem essentially unchanged may nonetheless ease its solution. Rescaling for example, may reduce numerical instability.

Karmarkar has observed that there is a transformation of the data more general than ordinary rescaling but equally natural. It occurs every time one views the feasible region of LPP at an oblique angle. The projection of the feasible region on one's retina is a distortion of the original problem; it is a special case of *projective transformation*. A projective transformation maps the lines to the lines but angles and distances change. As straight lines remain straight lines, while angles and distances change, under a projective transformation a polytope will remain a polytope but its orientation may change.

A key property of projective transformations is that a suitable one will move a point strictly inside a polytope to a place near the centre of the transformed polytope. One can verify this with Fig 11.4 by viewing it an angle and distance that makes $x^{(1)}$ appear to be at the centre of the polytope.

The Basic Strategy

The basic strategy of the projective scaling algorithm is as follows
(i) Take an interior point.
(ii) Transform the space so as to place the point near the centre of the polytope.
(iii) In the transformed space, move in the direction of the steepest descent but not all the way to the boundary of feasible region, so that the point remains in the interior of the transformed polytope. Taking inverse image, get the point in the original space.
(iv) Transform the space again to move this new point (in the original space) to a place near the centre of the polytope in the new transformed space and keep repeating the process until optimal solution has been obtained.

For performing above steps, we need the given LPP to be in a specified form which we call as Karmarkar's form, described below.

Linear Programming in Karmarkar's Form

Karmarkar, for his discussion assumed that the LPP be given in the following special form

$$\begin{aligned} \text{Min} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = 0 \\ & e^T x = 1 \\ & x \geq 0. \end{aligned} \tag{11.10}$$

Here $x \in \mathbf{R}^n, c \in \mathbf{R}^n, b \in \mathbf{R}^m, A = [a_{ij}]_{m\times n}$ and $e = (1,1,\ldots,1)^T \in \mathbf{R}^n$. The validity of Karmarkar's algorithm rests on the following two conditions on problem (11.10).

Assumption 1. The point $x = e/n = (1/n,\ldots,1/n)^T$ is feasible. (that is, $Ax = 0$ is satisfied by the point $x = e/n$; the other constraint $e^T x = 1$ is satisfied automatically for $x = e/n$)

Assumption 2. The minimum value of Karmarkar's Problem (11.10) is zero; that is, $Min \quad z = 0$.

There are various ways of putting a general LPP in the desired form so as to use Karmarkar's algorithm. The *centering transformation* does the trick for meeting Assumption 1. However, the transformation that gives Min $z = 0$ involves more tedious details, for example, using duality results of LPP and adjusting dual variables suitably. We shall explain this by an example in a later section. Before we discuss the centering transformation and other details of Karmarkar's algorithm, we shall like to visualize the feasible region of problem (11.10) geometrically and also the centre e/n of the polytope in the original space. For this we consider the following example

$$\begin{aligned} &\text{Min} && 3x_1 - x_2 + x_3 \\ &\text{subject to} && \\ & && -x_1 + x_3 = 0 \\ & && x_1 + x_2 + x_3 = 1 \\ & && x_1, x_2, x_3 \geq 0, \end{aligned}$$

and note that $e/3 = (1/3, 1/3, 1/3)^T$ is feasible for the given problem. Also the set $\{(x_1, x_2, x_3) : x_1 + x_2 + x_3 = 1\ , x_1, x_2, x_3 \geq 0\}$ is the outer face of the unit tetrahedron in $\mathbf{R}^3$ or to be precise the simplex S^2_x. Therefore the feasible region of the given problem is the intersection of the plane $-x_1 + x_3 = 0$ with the outerface of the tetrahedron as shown in Fig 11.5.

Geometrically the situation is very similar for problem (11.10). The set $\{x \in \mathbf{R}^n : e^T x = 1, x \geq 0\}$ is the simplex S^{n-1}_x and the constraint $Ax = 0$ represents a collection of m hyperplanes passing through the origin. Then the feasible region of problem (11.10) is the polytope which is obtained by taking the intersection of these hyperplanes with the simplex S^{n-1}_x.

11.6 Centering Transformation and Projection Matrix

In this section, we discuss the most novel and original idea of Karamaar's algorithm, namely, the *centering transformation*. The basic aim of the centering transformation is to map a *strictly feasible point* in the original space on to the center of the polytope in the transformed space. As this transformation is a projective transformation, a polytope in the original space will be mapped on to an another polytope in the transformed space having different orientation in general.

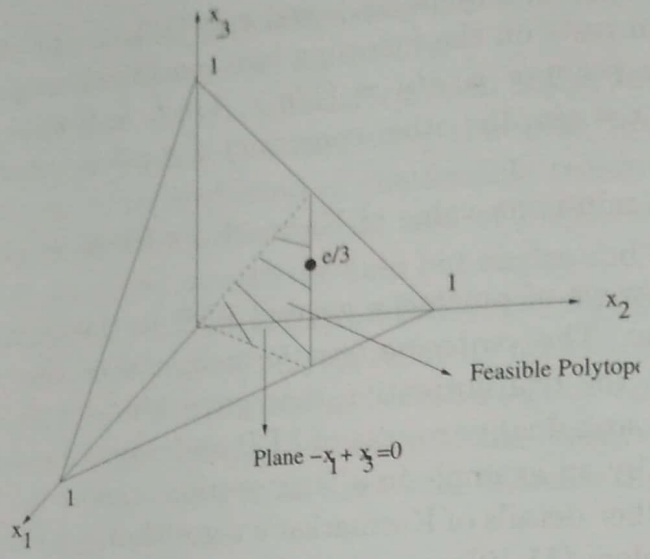


Fig. 11.5.

Suppose that the current feasible point $x^{(k)}$ is *strictly feasible* that is, all components of the vector $x^{(k)}$ are strictly positive. Karmarkar's *centering transformation* is a vector valued function $\theta_{x^{(k)}} : S_x^{n-1} \to S_y^{n-1}$ given by $\theta_{x^{(k)}}(x) = y$ where (i) $S_x^{n-1} = \{x \in \mathbf{R}^n : e^T x = 1, x \geq 0\} \subseteq \mathbf{R}^n$ is a $(n-1)$ simplex.

(ii) $y = \dfrac{D_k^{-1}x}{e^T D_k^{-1}x}$ for $x \in S_x^{n-1} \subset \mathbf{R}^n$, where $D_k = \text{diag}(x_1^{(k)}, \ldots, x_n^{(k)})$ with $D_k^{-1} = \text{diag}(1/x_1^{(k)}, \ldots, 1/x_n^{(k)})$, and $x^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})^T$ Here it must be noted that the mapping $\theta_{x^{(k)}}$ is defined with respect to the given strictly feasible point $x^{(k)}$. If the point $x^{(k)}$ is changed to a new strictly feasible point $x^{(k+1)}$ then θ will also change because the matrices D_k and D_k^{-1} will change to D_{k+1} and D_{k+1}^{-1}. Infact, this is the reason that we have used the notation $\theta_{x^{(k)}} : S_x^{n-1} \to S_y^{n-1}$ given by $\theta_{x^{(k)}}(x) = y$ for all $x \in S_x^{n-1}$. However, if there is no confusion then we shall omit the subscript $x^{(k)}$ and write $\theta_{x^{(k)}}$ as θ only.

Properties of the Centering Transformation

The centering transformation has the following properties

(i) The image of a strictly feasible point $x^{(k)}$ is the centre $\dfrac{e}{n}$ of the simplex S_y in the transformed space, i.e. $\theta(x^{(k)}) = \dfrac{D_k^{-1}x^{(k)}}{e^T D_k^{-1}x^{(k)}} = \dfrac{e}{n}$.

(ii) θ is one-to-one mapping of the simplex $S_x^{n-1} \subset \mathbf{R}^n$ onto the simplex $S_y^{n-1} \subset \mathbf{R}^n$. This is because for $\bar{x} \in S_x^{n-1}, \bar{y} = \theta(\bar{x}) = D_k^{-1}\bar{x} \big/ e^T D_k^{-1}\bar{x} > 0$ and $e^T\bar{y} = 1$. Also θ is invertible with $\theta^{-1}(y) = x = D_k y \big/ e^T D_k y$.

(iii) The set $\{x \in S_x^{n-1} : Ax = 0\}$ gets mapped to the set $\{y \in S_y^{n-1} : AD_k y = 0\}$ under

the mapping θ. This is because $Ax = 0$ gives $AD_k y / e^T D_k y = 0$ which is equivalent to $AD_k y = 0$. Therefore,

$$\{x \in S_x^{n-1} : Ax = 0\} \xrightarrow{\theta} \{y \in S_y^{n-1} : AD_k y = 0\} .$$

Thus a hyperplane passing through the origin shall be mapped to another hyperplane in the transformed space which also passes through the origin but has different orientation.

We illustrate some of these points with the help of the following example

$$\begin{aligned} \text{Min} \quad & 3x_1 - x_2 + x_3 \\ \text{subject to} \quad & \\ & -x_1 + x_3 = 0 \\ & x_1 + x_2 + x_3 = 1 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Here Fig. 11.6 shows the feasible region in the original space and the current point $x^{(k)} = (2/5, 2/5, 1/5)^T$ which is not the center $\frac{e}{3}$. Using the centering transformation θ, we get the new feasible region as shown in Fig. 11.6. Here $\theta(x^{(k)}) = \frac{e}{3}$ and the plane $-x_1 + x_3 = 0$ gets transformed to $-2/5y_1 + 1/5y_3 = 0$ under the mapping θ.

The Transformed Problem

Under the centering transformation, the original problem (11.10)

$$\begin{aligned} \text{Min} \quad & z = c^T x \\ \text{subject to} \quad & \\ & Ax = 0 \\ & e^T x = 1 \\ & x \geq 0, \end{aligned}$$

gets transformed into following linear fractional programming problem

$$\begin{aligned} \text{Min} \quad & \frac{c^T D_k y}{e^T D_k y} \\ \text{subject to} \quad & \\ & (AD_k)y = 0 \\ & e^T y = 1 \\ & y \geq 0, \end{aligned} \tag{11.11}$$

in the transformed space.

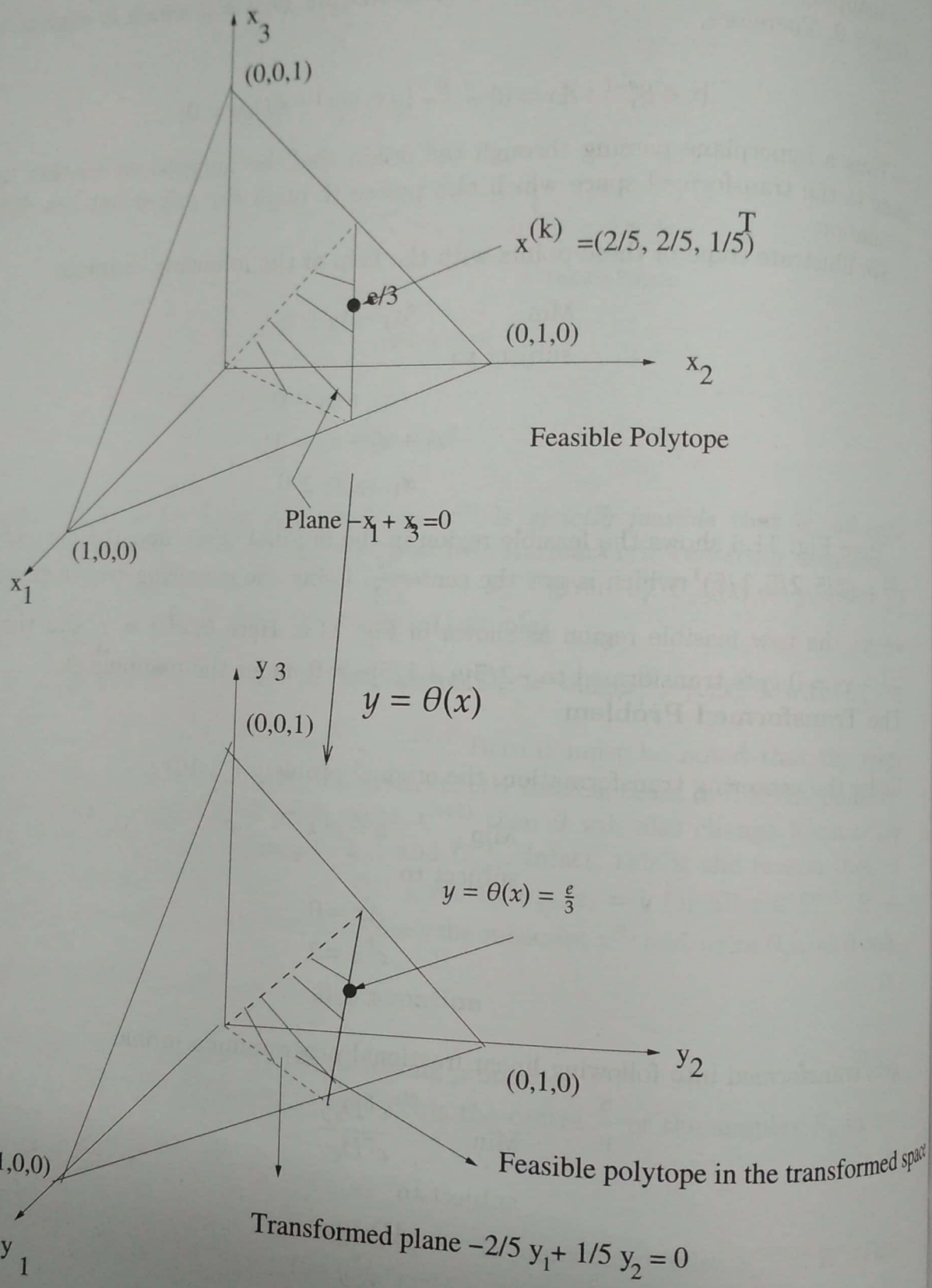


Fig. 11.6.

On the face of it, this situation does not seem to be very satisfactory because, in the transformed space problem (11.11) is no more a LPP. The objective function of (11.11), being the ratio of two linear functions, is not even convex (in fact it is *psuedo convex*, a class of generalized convex functions to be studied later in the book). But because of the Assumption 2 on problem (11.10), the minimum value of problem (11.11) is also zero. Also, as the denominator of the objective function in (11.11) is positive and bounded away from zero for all $y \in S_y^{n-1}$, problem (11.11) is equivalent to

$$\begin{aligned} \text{Min} \quad & (D_k c)^T y \\ \text{subject to} \quad & \\ & (AD_k)y = 0 \\ & e^T y = 1 \\ & y \geq 0. \end{aligned} \tag{11.12}$$

Now problem (11.12) is a LPP in the transformed space. In fact it is in the form of problem (11.11) with both the Assumptions 1 and 2 being satisfied.

Since, in the transformed space we shall be solving problem (11.12) and NOT problem (11.11), for problem (11.11), the sequence of iterates $\{x^{(k)}\}$ as generated by Karmarkar's algorithm approaches the optimal objective value zero in the limit as $k \to +\infty$, but not necessarily monotonically. Hence in practice, we terminate the algorithm when the objective function value of problem (11.11) gets sufficiently close to zero.

Projection Matrix

The next major step in Karmarkar's algorithm is to move from the centre $\frac{e}{n}$ in the direction of the negative gradient. Since, in general, the negative gradient may point in a direction away from the feasible region, we will need to find the projection of the negative gradient on the subspace given by all vectors $x \in \mathbf{R}^n$ for which $Ax = 0$ (or $(AD_k)y = 0$ in the transformed space). So we should obtain an expression for the projection matrix which should be used to find the desired projection. Though we have already learnt about the projection matrix in Chapter 10, we describe the same again for the sake of completeness.

Let A be a given $(m \times n)$ matrix and let $M = \{x \in \mathbf{R}^n : Ax = 0\}$ be the null space of A. Let $N = \{A^T w : w \in \mathbf{R}^m\}$ be the range space of A. Then $M \perp N$ and $\mathbf{R}^n = M \oplus N$; that is, any vector of $\mathbf{R}^n$ can be uniquely written as $u + v$ with $u \in M, v \in N$. Let $g_k = \nabla f(x_k)$ and $-g_k$ be the negative gradient. To find the direction of movement we have to project the negative gradient (that is, $-g_k$) onto the subspace of active constraints(that is, M). By above properties of direct sum we have

$$
\begin{aligned}
-g_k &= d_k + A^T w_k,\ d_k \in M \\
Ad_k &= -Ag_k - (AA^T)w_k = 0,\ A^T w_k \in N \\
w_k &= -(AA^T)^{-1}Ag_k \\
d_k &= -g_k - A^T w_k = -[I - A^T(AA^T)^{-1}A]g_k = -P_k g_k = P_k(-g_k)\,.
\end{aligned}
$$

Here $P_k = (I - A^T(AA^T)^{-1}A)$ is the *projection matrix.*

11.7 The Complete Algorithm

We now present the stepwise description of *Karmarkar's projective scaling algorithm.* Let the given LPP be in the form

$$
\begin{aligned}
\text{Min} \quad & z = c^T x \\
\text{subject to} \quad & \\
& Ax = 0 \\
& e^T x = 1 \\
& x \geq 0\,,
\end{aligned}
\tag{11.13}
$$

with the following two assumptions being satisfied, namely,

Step 1 (Initialization) Choose $x^{(0)} = e/n$ and set $k = 0$.
Step 2 (Stopping) If $c^T x^{(k)}$ is sufficiently close to zero, i.e. $c^T x^{(k)} < 2^{-L}$ then stop. In that case $x^{(k)}$ may be 'optimally rounded' to get an optimal b.f.s $\bar{x}$. Otherwise go to Step 3.
Step 3 (Projection) Compute the projection of the negative gradient of the transformed objective function on the feasible region of the transformed problem to get $c_p = (I - B^T(BB^T)^{-1}B)(D_k\, c)$, where $B = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}$.
Step 4 (New point) Update $y^{(k+1)} = \dfrac{e}{n} - \dfrac{\alpha}{\sqrt{(n(n-1))}} \dfrac{c}{\|c_p\|}$, here $\alpha \in (0,1)$ is a constant which is taken as 1/4 for convergence. Then compute

$$x^{(k+1)} = \theta^{-1}(y^{k+1}) = D_k\, y^{(k+1)} \Big/ e^T D_k\, y^{(k+1)}.$$

Step 5 Advance k to $(k+1)$ and continue.

Remark 11.7.1 *It is evident from the above steps that we always make a move from the center* $\dfrac{e}{n}$*, be it the original space or the transformed space. The direction of movement is always the projection of the negative gradient on to the subspace determined by the feasible set of the problem under consideration in the appropriate space. The step size is chosen as* $\dfrac{\alpha}{\sqrt{n(n-1)}}$ *because we want that the new point* $y^{(k+1)}$ *should remain a strictly*

feasible point, i.e. it should not hit the boundary of the polytope. This is guaranteed by taking $\alpha \in (0,1)$ *because* $\frac{1}{\sqrt{n(n-1)}}$ *is the radius of the 'largest' sphere which can be inscribed in an* $(n-1)$*-simplex. Some readers may like to verify this fact geometrically for the case of* $n = 3$.

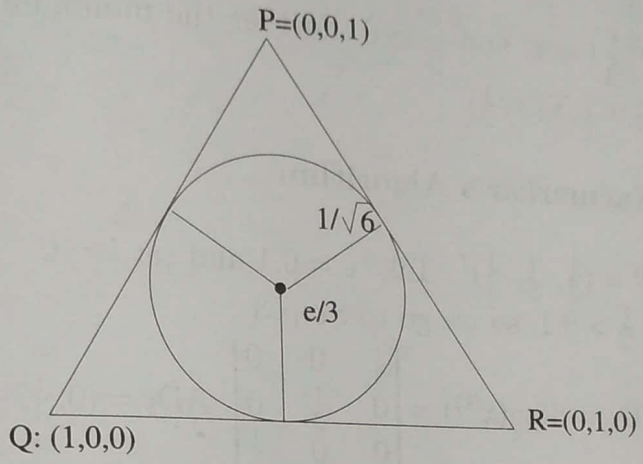


Fig. 11.7.

Remark 11.7.2 *As explained earlier in Section 11.6, since we are solving a LPP in the transformed space rather than the equivalent linear fractional programming problem, the algorithm does not guarantee a monotonic decrease in the objective function value of problem (11.13). However using the concept of potential function for both, the original and the transformed problems, Karmarkar proved that eventually the objective function value* $c^T x^{(k)}$ *for the current solution* $x^{(k)}$ *will tend to zero as* $k \to +\infty$.

Remark 11.7.3 *Since initially as well as for the subsequent iterations, the iterates* $x^{(k)}$ *are strictly feasible, the point* $x^{(p)}$ *at which we stop will still remain in the interior of the feasible region. But as the optimal solution* $\bar{x}$ *of a LPP is a corner point, we must have a procedure which takes us from* $x^{(p)}$ *to* $\bar{x}$*, again in polynomial time. This procedure is called the 'purification step' or 'optimal rounding scheme' and that we discuss after illustrating the working of Karmarkar's algorithm for a small example.*

Example 11.7.1 *Use Karmarkar's algorithm to solve the following LPP*

$$\text{Min} \quad z = x_1 + 3x_2 - 3x_3$$

subject to

$$\begin{aligned} x_2 - x_3 &= 0 \\ x_1 + x_2 + x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Solution Before we start solving the given LPP by Karmarkar's algorithm, we have to make sure that the given LPP is in the form of problem (11.13) with both assumptions being satisfied.

Looking at the given LPP, we note that by taking $x = (x_1, x_2, x_3)^T$, $A = [0, 1, -1]$, $c = (1, 3, -3)^T$ the problem is in the desired form. Also $\frac{e}{3}$ is feasible for the given problem because $A\left(\frac{e}{3}\right) = 0$, $e^T(\frac{e}{3}) = 1$, and $\frac{e}{3} \geq 0$. Further the minimum value is zero which is attained at $(\overline{x_1} = 0, \overline{x_2} = \frac{1}{2}, \overline{x_3} = \frac{1}{2})$.

First iteration of Karmarkar's Algorithm

Step 1 Start with $x^{(0)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$. Take $\epsilon = 0.1$ and set $k = 0$.
Step 2 $x^{(k)}$ yields $z = \frac{1}{3} > 0.1$, so we go to Step 3.

Step 3 $A = (0, 1, -1)$, $D_k = \text{diag}(x^{(k)}) = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$, $AD_k = (0, \frac{1}{3}, -\frac{1}{3})$

$$B = \begin{bmatrix} AD_k \\ e \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & -\frac{1}{3} \\ 1 & 1 & 1 \end{bmatrix}, \quad BB^T = \begin{bmatrix} \frac{2}{9} & 0 \\ 0 & 3 \end{bmatrix}, \quad (BB^T)^{-1} = \begin{bmatrix} \frac{9}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$

$$P = I - B^T(BB^T)^{-1}B = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

$$c = (1, 3, -3)^T, \; D_k c = \left(\frac{1}{3}, 1, -1\right)^T$$

$$P(D_k c) = c_p = \left(\frac{2}{9}, -\frac{1}{9}, -\frac{1}{9}\right)^T.$$

Using $\alpha = \frac{1}{4}$ we get

$$y^{(1)} = \frac{e}{n} - \frac{\alpha}{\sqrt{n(n-1)}} \frac{c_p}{\|c_p\|} = \left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right)^T$$

and

$$x^{(1)} = \theta^{-1}(y^{(1)}) = \frac{D_k y^{(1)}}{e^T D_k y^{(1)}} = \left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right)^T.$$

Here we note that as the starting point is $\frac{e}{3}$, $x^{(1)}$ will be same as $y^{(1)}$. This will not happen if the starting point is different from $\frac{e}{3}$. Also

$$c^T x^{(1)} = \frac{1}{4} + 3\left(\frac{3}{8}\right) = \frac{1}{4} < \frac{1}{3}.$$

Second iteration of Karmarkar's Algorithm

$$A = (0, 1, -1), \quad D_k = \operatorname{diag}(x^{(k)}) = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{3}{8} & 0 \\ 0 & 0 & \frac{3}{8} \end{bmatrix}, \; AD_k = (0, \tfrac{3}{8}, -\tfrac{3}{8}),$$

$$B = \begin{bmatrix} AD_k \\ e \end{bmatrix} = \begin{bmatrix} 0 & \frac{3}{8} & -\frac{3}{8} \\ 1 & 1 & 1 \end{bmatrix}, \quad BB^T = \begin{bmatrix} \frac{9}{32} & 0 \\ 0 & 3 \end{bmatrix}, \quad (BB^T)^{-1} = \begin{bmatrix} \frac{32}{9} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$

$$P = I - B^T(BB^T)^{-1}B = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

$$c = (1, 3, -3)^T, D_k c = \left(\frac{1}{4}, \frac{9}{8}, -\frac{9}{8}\right)^T$$

$$P(D_k c) = c_p = \left(\frac{1}{6}, -\frac{1}{12}, -\frac{1}{12}\right)^T.$$

Using $\alpha = \frac{1}{4}$ we get

$$y^{(2)} = \frac{e}{n} - \frac{\alpha}{\sqrt{n(n-1)}} \frac{c_p}{\|c_p\|} = \left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right)^T,$$

and

$$x^{(2)} = \theta^{-1}(y^{(2)}) = \frac{D_k y^{(2)}}{e^T D_k y^{(2)}} = \left(\frac{2}{11}, \frac{9}{22}, \frac{9}{22}\right)^T.$$

Also

$$c^T x^{(2)} = \frac{2}{11} + 3\left(\frac{9}{22}\right) - 3\left(\frac{9}{22}\right) = \frac{2}{11} = 0.\overline{18}.$$

Thus we have

$$x^{(0)} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), c^T x^{(0)} = \frac{1}{3} = 0.\bar{3}$$

$$x^{(1)} = \left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right), c^T x^{(1)} = \frac{1}{4} = 0.25$$

$$x^{(2)} = \left(\frac{2}{11}, \frac{9}{22}, \frac{9}{22}\right), c^T x^{(2)} = \frac{2}{11} = 0.\overline{18}$$

Purification Step or Optimal Rounding Scheme

Here we are given the final iterate $x^{(p)}$ such that $c^T x^{(p)} < 2^{-L}$. The purification step or the optimal rounding scheme determines a corner point (b.f.s.) which gives at least the objective function value $c^T x^{(p)}$.

If n linearly independent constraints are binding at $x^{(p)}$, then it is already a basic feasible solution. Otherwise, we can find a direction $d \in \mathbf{R}^n$, $(d \neq 0)$ lying in the null space of binding constraints, i.e. satisfying homogeneous system of equality constraints corresponding to the binding constraints. We now move from the point $x^{(p)}$ in the direction $-d$ if $c^T d < 0$ and along the direction d if $c^T d > 0$, until some constraint blocks further movement because of feasibility restriction. This has to happen as the feasible

region is bounded. Thus we obtain a new solution $x^{(p+1)}$ such that $c^T x^{(p+1)} < 2^{-L}$ and at least one additional linearly independent constraint is binding. Proceeding in this manner, a basic feasible solution $\bar{x}$ to problem (11.13) can be obtained such that $c^T\bar{x} < 2^{-L}$. Here we note that the process involves $n-(m+1)$ such steps since it begins with $(m+1)$ linearly independent equality constraints (binding) in problem (11.13) binding and then adds at least one additional linearly independent binding constraint to this set at each step. Also each step is of polynomial complexity. Hence $\bar{x}$ is determined from the final iterate $x^{(p)}$ in polynomial time.

In our example (Example 11.7.1) let the final iterate be $x^{(2)}$. We observe that only two linearly independent equality constraints are binding at $x^{(2)}$ (as $Ax^{(2)} = 0$, $e^T x^{(2)} = 1$). Hence we determine $d = (d_1, d_2, d_3)^T \neq 0$ such that $d_2 - d_3 = 0, d_1 + d_2 + d_3 = 0$. this gives $d_2 = d_3$ and $d_1 = -2d_3$. Taking $d_3 = 1$ we get $d = (-2,1,1)^T$. Since $c^T d = d_1 + 3d_2 - 3d_3 = d_1 = -2 < 0$ we move along the direction of $d = (-2,1,1)^T$, i.e. $\bar{x} = x^{(2)} + \alpha d = \left(\frac{2}{11}, \frac{9}{22}, \frac{9}{22}\right) + \alpha(-2,1,1) = \left(\frac{2}{11} - 2\alpha, \frac{9}{22} + \alpha, \frac{9}{22} + \alpha\right)$. As $\frac{2}{11} - 2\alpha \geq 0$, we get $\alpha \leq \frac{1}{11}$. Therefore for $\alpha = \frac{1}{11}$, the constraint $x_1 \geq 0$ blocks the motion and it becomes the third (linearly independent) equality constraint which is binding. Hence we get $\bar{x} = \left(\frac{2}{11} - \frac{2}{11}, \frac{9}{22} + \frac{1}{11}, \frac{9}{22} + \frac{1}{11}\right) = \left(0, \frac{1}{2}, \frac{1}{2}\right)$ is the required corner point solution.

11.8 Putting a general LPP in Karmarkar's form

We shall illustrate this by an example only. Let the given LPP be

$$\begin{aligned} \text{Max} \quad & z = 3x_1 + x_2 \\ \text{subject to} \quad & \\ & 2x_1 - x_2 \leq 2 \\ & x_1 + 2x_2 \leq 5 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{11.14}$$

To put problem (11.14) in Karmarkar's form (i.e. in the form of problem (11.13)) we perform the following steps

Step 1 Write the dual of (11.14), i.e.

$$\begin{aligned} \text{Min} \quad & w = 2y_1 + 5y_2 \\ \text{subject to} \quad & \\ & 2y_1 + y_2 \geq 3 \\ & -y_1 + 2y_2 \geq 1 \\ & y_1, y_2 \geq 0. \end{aligned} \tag{11.15}$$

Step 2 From duality theorem, any feasible solution of the following set of constraints will yield the optimal solution of (11.14)

$$\begin{aligned} 3x_1 + x_2 - 2y_1 - 5y_2 &= 0 \\ 2x_1 - x_2 &\le 2 \\ x_1 + 2x_2 &\le 5 \\ 2y_1 + y_2 &\ge 3 \\ -y_1 + 2y_2 &\ge 1 \\ x_1, x_2, y_1, y_2 &\ge 0. \end{aligned} \tag{11.16}$$

Step 3 Introduce slack and surplus variables in the system (11.16) to yield

$$\begin{aligned} 3x_1 + x_2 - 2y_1 - 5y_2 &= 0 \\ 2x_1 - x_2 + s_1 &= 2 \\ x_1 + 2x_2 + s_2 &= 5 \\ 2y_1 + y_2 - r_1 &= 3 \\ -y_1 + 2y_2 - r_2 &= 1 \\ x_1, x_2, y_1, y_2, r_1, r_2, s_1, s_2 &\ge 0. \end{aligned} \tag{11.17}$$

Step 4 Find a number M such that any feasible solution to (11.17) will satisfy $\sum x_i + \sum y_j + \sum s_i + \sum r_j \le M$ and append this constraint to (11.17). In our example if we assume that all variables have an upper bound of 10, then $x_1 + x_2 + y_1 + y_2 + s_1 + s_2 + r_1 + r_2 \le (10 \times 8) = 80$. We then add a slack variable (dummy variable d_1) to this constraint. Our new goal is then to find a feasible solution of

$$\begin{aligned} 3x_1 + x_2 - 2y_1 - 5y_2 &= 0 \\ 2x_1 - x_2 + s_1 &= 2 \\ x_1 + 2x_2 + s_2 &= 5 \\ 2y_1 + y_2 - r_1 &= 3 \\ -y_1 + 2y_2 - r_2 &= 1 \\ x_1 + x_2 + y_1 + y_2 + s_1 + s_2 + r_1 + r_2 + d_1 &= 80 \\ x_1, x_2, y_1, y_2, r_1, r_2, s_1, s_2, d_1 &\ge 0. \end{aligned} \tag{11.18}$$

Step 5 Now define a new dummy variable d_2 such that $d_2 = 1$. We use this variable d_2 (which is equal to 1) to make the R.H.S of (11.18) (except the last constraint) equal to zero. To do this we add the appropriate multiple of the constraint $d_2 = 1$ to each constraint of (11.18) (except the last constraint) having a non zero R.H.S. (e.g. we add $-2(d_2 = 1)$ to the constraint $2x_1 - x_2 + s_1 = 2$). We also replace the last constraint in (11.18) by the following two constraint

(i) Add $d_2 = 1$ to the last constraint
(ii) Subtract $M(=80)$ times $(d_2 = 1)$ from the last constraint of (11.18).

We now seek a feasible solution of

$$\begin{aligned}
3x_1 + x_2 - 2y_1 - 5y_2 &= 0\\
2x_1 - x_2 + s_1 - 2d_2 &= 0\\
x_1 + 2x_2 + s_2 - 5d_2 &= 0\\
2y_1 + y_2 - r_1 - 3d_2 &= 0\\
-y_1 + 2y_2 - r_2 - d_2 &= 0\\
x_1 + x_2 + y_1 + y_2 + s_1 + s_2 + r_1 + r_2 + d_1 - 80d_2 &= 0\\
x_1 + x_2 + y_1 + y_2 + s_1 + s_2 + r_1 + r_2 + d_1 + d_2 &= 81\\
x_1, x_2, y_1, y_2, r_1, r_2, s_1, s_2, d_1, d_2 &\geq 0. \qquad (11.19)
\end{aligned}$$

Here $d_2 = 1$ and $x_1 + x_2 + y_1 + y_2 + s_1 + s_2 + r_1 + r_2 + d_1 = 80$, are equivalent to the last two constraint (11.19).

Step 6 Make the following change of variables
$x_j = (M+1)x'_j$, $y_j = (M+1)y'_j$, $s_j = (M+1)s'_j$, $r_j = (M+1)r'_j$, $d_j = (M+1)d'_j$ $(j = 1, 2)$.
This yields

$$\begin{aligned}
3x'_1 + x'_2 - 2y'_1 - 5y'_2 &= 0\\
2x'_1 - x'_2 + s'_1 - 2d'_2 &= 0\\
x'_1 + 2x'_2 + s'_2 - 5d'_2 &= 0\\
2y'_1 + y'_2 - r'_1 - 3d'_2 &= 0\\
-y'_1 + 2y'_2 - r'_2 - d'_2 &= 0\\
x'_1 + x'_2 + y'_1 + y'_2 + s'_1 + s'_2 + r'_1 + r'_2 + d'_1 - 80d'_2 &= 0\\
x'_1 + x'_2 + y'_1 + y'_2 + s'_1 + s'_2 + r'_1 + r'_2 + d'_1 + d'_2 &= 81\\
x'_1, x'_2, y'_1, y'_2, r'_1, r'_2, s'_1, s'_2, d'_1, d'_2 &\geq 0. \qquad (11.20)
\end{aligned}$$

We now ensure that a point that sets all variables equal is feasible to (11.20). For this, we first add a dummy variable d'_3 (artificial variable) to the last constraint in (11.20) and then add a multiple of d'_3 to each of other constraint. This multiple is chosen so that sum of the coefficient of all variables in each constraint (except the last) will be equal to zero. This yields the following LPP.

Step 7

Min $z = d'_3$

subject to

$$\begin{aligned}
3x'_1 + x'_2 - 2y'_1 - 5y'_2 + 3d'_3 &= 0\\
2x'_1 - x'_2 + s'_1 - 2d'_2 + d'_3 &= 0\\
x'_1 + 2x'_2 + s'_2 - 5d'_2 + d'_3 &= 0\\
2y'_1 + y'_2 - r'_1 - 3d'_2 + d'_3 &= 0\\
-y'_1 + 2y'_2 - r'_2 - d'_2 + d'_3 &= 0\\
x'_1 + x'_2 + y'_1 + y'_2 + s'_1 + s'_2 + r'_1 + r'_2 + d'_1 - 80d'_2 + 71d'_3 &= 0\\
x'_1 + x'_2 + y'_1 + y'_2 + s'_1 + s'_2 + r'_1 + r'_2 + d'_1 + d'_2 + d'_3 &= 1\\
x'_1, x'_2, y'_1, y'_2, r'_1, r'_2, s'_1, s'_2, d'_1, d'_2, d'_3 &\geq 0. \qquad (11.21)
\end{aligned}$$

In (11.21), the point $x'_1 = x'_2 = y'_1 = y'_2 = r'_1 = r'_2 = s'_1 = s'_2 = d'_1 = d'_2 = d'_3 = \frac{e}{n} = \frac{1}{11}$ is feasible. Since d'_3 should be zero in a feasible solution of (11.20) we need to take the objective function as *Min* d'_3 in (11.21). If (11.21) is feasible then *Min* $d'_3 = 0$ and then values of remaining variables will yield a feasible solution of (11.14) and hence the values of x_1 and x_2 in the optimal solution of (11.21) (with *Min* $d'_3 = 0$) will give a optimal solution of the given LPP (11.14). Then LPP in (11.21) satisfies all conditions of Karamaker's form (11.13) and hence is ready to be solved by Karmarkar's algorithm.

11.9 Worst Case Computational Complexity of Karmarkar's Algorithm

We shall now show that the computational complexity of Karmarkar's algorithm is $O(n^{3.5}L)$. For this we shall make use of the following Lemma due to Karmarkar.

Lemma 11.9.1 *Let $x^{(k)}, c, \alpha$, and $y^{(k+1)}$ be as in the description of Karmarkar's algorithm. Then either*

(i) $(D^k c)y^{(k+1)} = 0$, *or*

(ii) $g\left(\frac{e}{n}\right) - g(y^{(k+1)}) \geq \delta$

where $D^k = diag(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ *and δ is a constant (> 0), depending on α. For $\alpha = 1/4$, $\delta \geq 1/8$.*

Here g is the potential function given by

$$g(y) = \sum_{i=1}^{n} \ln\left(\frac{(D_k c)y}{y_j}\right) - \sum_{j=1}^{n} \ln(x_j^{(k)}),$$

for the transformed cost and $f(x) = \sum_{j=1}^{n} \ln\left(c^T x / x_j^{(k)}\right)$ *is the potential function for the true cost.*

Theorem 11.9.1 *Any linear programming problem can be polynomially reduced to one in the form of problem (11.13) specified for Karmarkar's Algorithm. Moreover in time polynomial in the size of the input, the algorithm either computes a solution or proves that none exist.*

Proof. We have already shown how a general LPP is reduced to Karmarkar's Algorithm format. For proving polynomial time complexity of Karmarkar's algorithm, we consider the implications of two cases as given in the Lemma 11.9.1.

Case(i) $(D_k c)y^{(k+1)} = 0$. This gives

$$c^T x^{(k+1)} = \frac{\sum_{j=1}^{n} c_j x_j^k y_j^{(k+1)}}{\sum_{j=1}^{n} x_j^k y_j} = 0,$$

and therefore we stop.

Case(ii) $g(\frac{e}{n}) - g(y^{(k+1)}) \geq \delta$. Then the relation $f(x) = g(\theta(x))$ assures that

$$g(\theta(x^{(k)})) - g(\theta(x^{(k+1)})) = f(x^{(k)}) - f(x^{(k+1)}) \geq \delta\,. \tag{11.22}$$

Therefore, (11.22) gives

$$f(x^{(k+1)}) \leq f(x^{(k)}) - (k+1)\delta, \tag{11.23}$$

or

$$f(x^{(k)}) \leq f(x^{(k-1)}) - \delta \leq \ldots \leq f(x^{(0)}) - k\delta. \tag{11.24}$$

If at any iteration condition (i) arises then the algorithm will clearly stop as the minimum value of problem (11.13) is assumed to be zero at Step 1. It runs k iterations without stopping in that way, then (11.24) yields

$$f(x^{(k)}) \leq f(x^{(0)}) - k\delta. \tag{11.25}$$

Now applying the definition of f , (11.25) gives

$$\sum_{j=1}^{n} \ln\left(\frac{c^T x^{(k)}}{x_j^{(k)}}\right) \leq \sum_{j=1}^{n} \ln\left(\frac{c^T x^{(0)}}{x_j^0}\right) - k\delta,$$

or

$$n \ln\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq \sum_{j=1}^{n} \ln(x_j^{(k)}) - \sum_{j=1}^{n} \ln(x_j^{(0)}) - k\delta\,. \tag{11.26}$$

But $x_j^{(0)} = \frac{1}{n}$, as $x^{(0)} = \frac{e}{n}$ for all j Therefore from (11.26)

$$n \ln\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq \sum_{j=1}^{n} \ln(x_j^{(k)}) - k\delta \,. \tag{11.27}$$

Further $x_j^{(k)} \leq 1$ for all j as $x^{(k)}$ is on the simplex S_x^{n-1}, the equation (11.27) gives

$$n \ln\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq n \ln n - k\delta. \tag{11.28}$$

Now choosing $\bar{k} = \dfrac{nq + n \ln n}{\delta}$, where q is the tolerance parameter, we have from (11.28)

$$n \ln\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq n \ln n - \frac{nq + n \ln n}{\delta}\delta,$$

or

$$n \ln\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq -nq \,.$$

Therefore,

$$\left(\frac{c^T x^{(k)}}{c^T x^{(0)}}\right) \leq \exp(-q) \leq 2^{-q} \text{ for } \quad q > 0.$$

Hence, after $\bar{k} = O(nq + n \ln n)$ iterations, the algorithm will stop at Step 2. Since each iteration requires $O(n^{2.5})$ computation to do necessary projections (using rank 1 update rule), the entire computation is $O(n^{3.5}q + n^{3.5} \ln n)$ steps, which can be expressed as $O(n^{3.5}(L + \ln n))$ where L is the length of the input.

11.10 Summary and Additional Notes

- Certain complexity issues with regard to the simplex algorithm and an introductory discussion on Karmarkar's projective scaling algorithm constitute the core of this chapter.
- Section 11.4 is devoted to the study of Klee and Minty cube and its variant so as to show that the simplex algorithm is not a polynomial time algorithm.
- Section 11.5 to 11.7 are devoted to the study of various aspects of Karmarkar's algorithm. While Karmarkar's centering transformation is presented in Section 11.7, a complete description of the main algorithm is given in Section 11.8. Section 11.9 describes a procedure based on duality so as to transform the given LPP in Karmarkar's format. The main theoretical results that the worst case computational complexity of Karmarkar's algorithm is polynomial, is presented in Section 11.9.

- The Soviet mathematician L. G. Khachian was the first to develope a polynomial time algorithm for solving LPP's in 1979. Khachian's algorithm (named as Ellipsoid algorithm) derives its main idea from an earlier work of another Soviet mathematicians N.Shor.
- The projective scaling algorithm was developed by Narendra Karmarkar in 1984 at AT and T Bell Labs (USA). Though Khachian's algorithm and Karmarkar's algorithm both exhibit that the class of linear programming problems is in the class P of problems, Khachian's algorithm did not prove to be of significant practical computational value. This was because the ellipsoid method requires that calculations be done with very high precision. Karmarkar's algorithm is free of the ellipsoids precision problem, and its worst case behaviour is also substantially better than Khachian's algorithm.
- From the very inseption, there have been many numerical experiments on LPP's of various sizes so as to compare Karmarkar's algorithm and Dantzig's simplex method. It has been observed that for LPP's of moderate size, the simplex method performs quite well in practise, despite its dreadful worst case behaviour. The main reason for this could be the fact that the *average case probabilistic computational complexity* of the simplex algorithm is polynomial (Borgwardt [24]).
- Although there is no evidence to show that the projective scaling algorithm can beat the simplex algorithm by a factor of 50, (as originally claimed by Karmarkar) there is consensus that the number of iterations in Karmarkar's algorithm grows very slowly with problem size as predicted by Karmarkar. But each iteration requires a nasty least squares problem to be solved to determine the projection matrix. There have been various modifications in the original Karmarkar's algorithm to speed up by employing an efficient least squares routine. With these modifications, Karmarkar's algorithm has proved to be very efficient for solving LPP's of very large size.
- Various authors proposed *affine scaling algorithms* as variants of the original Karmarkar's algorithm. As with Karmarkar's original approach, the affine scaling algorithms are also interior point algorithms. The added advantage of these algorithms is the fact that they can be applied directly to LPP's in the standard form, unlike Karmarkar's algorithm which requires a very specific form of the given LPP. Though the basic affine scaling algorithm (both in primal and dual forms) is straight forward, it could not be proved to be polynomial time without incorporating additional details. An excellent book for various interior point methods for solving LPP's is Fang and Puthenpura [54], which also discusses the determination of dual variables etc.

11.11 Exercises

11.1 *Consider the LPP*

$$\begin{aligned} &\textit{Min} && x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 - 2x_3 = 0 \\ & && x_1 + x_2 + x_3 = 1 \\ & && x_1,\ x_2,\ x_3 \geq 0\,. \end{aligned}$$

Can the above LPP be solved directly by Karmarkar's algorithm? Give reason for your answer. Hence perform one complete iteration of the said algorithm.

11.2 *Consider the LPP*

$$\begin{aligned} &\textit{Min} && -x_1 - 2x_2 + 4x_5 \\ &\textit{subject to} \\ & && x_1 - x_2 + 2x_3 - x_5 = 0 \\ & && x_1 + 2x_2 + x_4 - 4x_5 = 0 \\ & && x_1 + x_2 + x_3 + x_4 + x_5 = 1 \\ & && x_1,\ x_2,\ x_3,\ x_4\ x_5 \geq 0\,. \end{aligned}$$

Check for the above LPP

1. $x^{(0)} = e/5$ *is feasible.*
2. $x^* = (0, 2/5, 2/5, 0, 1/5)^T$ *is optimal.*
3. *The optimal value is zero. Hence perform one complete iteration of Karmarkar's algorithm.*

11.3 *Consider the LPP*

$$\begin{aligned} &\textit{Max} && 2x_1 + x_2 \\ &\textit{subject to} \\ & && x_1 - x_2 \leq 2 \\ & && x_1 + 2x_2 \leq 4 \\ & && x_1,\ x_2 \geq 0\,. \end{aligned}$$

Use duality to convert the above LPP in the form (KLP) so that Karmarkar's algorithm could be used.

11.4 *Determine the size of the LPP given at Question 11.3 above. Obtain the isotonic path of maximum length and hence determine the maximum number of iterations (in the worst case) needed to get its optimal solution if it has to be solved by the simplex algorithm.*

11.5 *Consider the LPP*

$$\begin{aligned} &\textit{Max} \quad 2x_1 - x_2 \\ &\textit{subject to} \\ &\qquad x_1 - x_2 + x_3 = 0 \\ &\qquad x_1 + x_2 + x_3 = 1 \\ &\qquad x_1,\ x_2,\ x_3 \geq 0\ . \end{aligned}$$

Is the point $\bar{x} = (1/4, 1/2, 1/4)^T$ *strictly feasible? Let* $\theta_{\bar{x}}$ *be the centering transformation with respect to the given point* $\bar{x}$. *Obtain*

1. $\theta_{\bar{x}}(1/4, 1/2, 1/4)$
2. $\theta_{\bar{x}}(1/3, 1/3, 1/3)$
3. $\theta_{\bar{x}}(0, 0, 1)$
4. $\theta_{\bar{x}}(1/2, 1/2, 0)$
5. $\theta_{\bar{x}}(1/2, 1/4, 1/4)$.

11.6 *Consider the LPP*

$$\begin{aligned} &\textit{Max} \quad x_2 \\ &\textit{subject to} \\ &\qquad x_1 + x_2 - x_3 = 0 \\ &\qquad x_1 + x_2 + x_3 = 1 \\ &\qquad x_1,\ x_2,\ x_3 \geq 0\ . \end{aligned}$$

1. *Is* $x^{(0)} = (1/4, 1/4, 1/2)^T$ *strictly feasible?*
2. *Use the explicit representation of the mapping* $\theta_{x^{(0)}} : S_x^2 \to S_y^2$ *to obtain the problem to be solved in the transformed space.*
3. *Starting with* $x^{(0)}$, *perform one complete iteration of Karmarkar's algorithm.*

11.7 *Consider the LPP*

$$\begin{aligned} &\textit{Max} \quad 6x_1 - 6x_2 + x_3 \\ &\textit{subject to} \\ &\qquad x_1 - x_2 = 0 \\ &\qquad x_1 + x_2 + x_3 = 1 \\ &\qquad x_1,\ x_2,\ x_3 \geq 0\ . \end{aligned}$$

1. *Show that the given LPP meets all assumptions required to apply Karmarkar's algorithm.*
2. *Starting with* $x^{(0)} = (1/4, 1/4, 1/2)^T$, *perform one complete iteration of Karmarkar's algorithm.*
3. *Sketch the feasible region of the original as well as the transformed problem. Hence indicate the movement of the algorithm in the original space.*

11.8 *Consider the LPP*

$$\begin{aligned} &\textit{Max} && 2x_1 + 3x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 \le 8 \\ & && 2x_1 + x_2 \le 10 \\ & && x_1,\ x_2 \ge 0\ . \end{aligned}$$

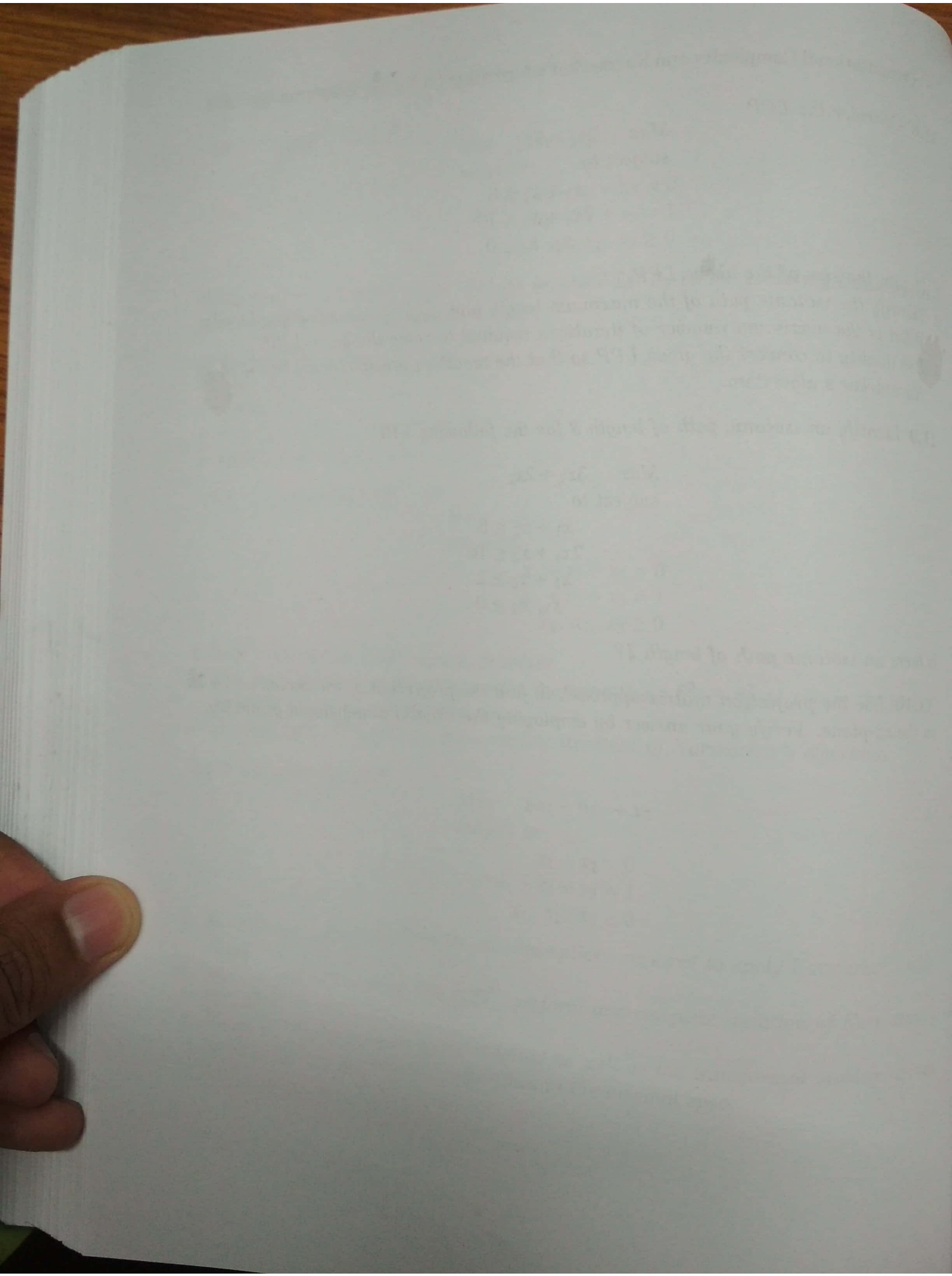
1. *Obtain the size of the above LPP.*
2. *Identify the isotonic path of the maximum length and indicate the same graphically.*
3. *What is the maximum number of iterations required to solve the given LPP.*
4. *Use duality to convert the given LPP so that the resulting problem could be solved by Karmarkar's algorithm.*

11.9 *Identify an isotonic path of length 3 for the following LPP*

$$\begin{aligned} &\textit{Max} && 3x_1 + 2x_2 \\ &\textit{subject to} \\ & && x_1 + x_2 \le 8 \\ & && 2x_1 + x_2 \le 10 \\ & && x_1 + x_2 \ge 2 \\ & && x_1,\ x_2 \ge 0\ . \end{aligned}$$

Is there an isotonic path of length 4?

11.10 *Use the projection matrix approach to find the projection of the vector* $\hat{i} - \hat{j} + 2\hat{k}$ *on the xz-plane. Verify your answer by employing the usual 3 dimensional geometry.*



12
Some Generalized Convex Functions and Fractional Programming

12.1 Introduction

By now we must have realized the potential of convexity in nonlinear optimization. We have already seen that properties of convex functions play pivotal role in the theoretical and algorithmic developments, in the last few chapters. Some of these properties have been studied in Chapter 7. So, it is natural for us to ask, what if we move away from convexity? Can the properties of convex functions be extended to some other classes of functions? In this chapter we attempt to answer such questions in affirmative. Beginning with the introduction of four new classes of functions, namely, quasiconvex functions, strictly quasiconvex functions, pseudoconvex functions and strictly pseudoconvex functions, that can be viewed as generalizations of the class of convex functions, we move on to describe their applications in fractional programming problems.

12.2 Quasiconvex and Quasiconcave Functions

One of the important property of convex function is that its level sets are convex sets, i.e. if $S \subseteq \mathbf{R}^n$ is a convex set and $f : S \to \mathbf{R}$ is a convex function on S then the level sets of f, given by $\Gamma_\alpha = \{x \in S : f(x) \leq \alpha\}$, are convex sets, for every $\alpha \in \mathbf{R}$. It was noted in Chapter 7 that the converse of this statement is not necessarily true. For instance, the function $f(x) = x^3$, $x \in \mathbf{R}$, is obviously a nonconvex function but its level sets are intervals, $(-\infty, |\alpha|^{1/3}]$, which are convex sets, for any $\alpha \in \mathbf{R}$. We look at a little more involved example, say, $f : [0,2] \to \mathbf{R}$ defined by

$$f(x) = \begin{cases} \sqrt{1-x^2}, & 0 \leq x \leq 1 \\ \sqrt{1-(x-2)^2}, & 1 \leq x \leq 2. \end{cases}$$

Then f is not a convex function on $[0,2]$ (draw the graph of f to see it). The level sets of f,

$$\Gamma_\alpha = \begin{cases} \emptyset, & \alpha < 0 \\ \left[\sqrt{1-\alpha^2},\ 2-\sqrt{1-\alpha^2}\right], & 0 \le \alpha \le 1 \\ \mathbf{R}, & \alpha > 1, \end{cases}$$

are convex sets.

Thus, we look for a new class of functions that can be completely characterized in terms of the level sets. This leads to the notion of quasiconvexity.

Definition 12.2.1 (Quasiconvex Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a quasiconvex function on S if for all $x, u \in S$ and for all $0 \le \lambda \le 1$,*

$$f(\lambda x + (1-\lambda)u) \le Max\{f(x), f(u)\}.$$

Analogously, we can define quasiconcave function.

Definition 12.2.2 (Quasiconcave Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a quasiconcave function on S if for all $x, u \in S$ and for all $0 \le \lambda \le 1$,*

$$f(\lambda x + (1-\lambda)u) \ge Min\{f(x), f(u)\}.$$

From the above two definitions, we can observe the following

(i) f is a quasiconcave function at u if and only if $-f$ is a quasiconvex function at u.

(ii) Every convex (respectively, concave) function is quasiconvex (respectively, quasiconcave), but the converse is not necessarily true. We will see few examples later to support this assertion.

(iii) A quasiconvex (respectively, quasiconcave) function need not be continuous in its domain. For example, consider a function $f(x) = [x]$, where $[x]$ is the greatest integer function not greater than x. Then f is a quasiconvex function but fails to be continuous at integer points.

(iv) The sum of two quasiconvex (respectively, quasiconcave) functions need not be a quasiconvex (respectively, quasiconcave) function. For example, take $f_1, f_2 : \mathbf{R} \to \mathbf{R}$ as

$$f_1(x) = \begin{cases} |x-1|-1, & 0 \le x \le 2 \\ 0, & \text{otherwise;} \end{cases} \qquad f_2(x) = \begin{cases} |x-3|-1, & 2 \le x \le 4 \\ 0, & \text{otherwise.} \end{cases}$$

Then,

$$f(x) = f_1(x) + f_2(x) = \begin{cases} |x-1|-1, & 0 \le x \le 2 \\ |x-3|-1, & 2 \le x \le 4 \\ 0 & \text{otherwise.} \end{cases}$$

Now both f_1 and f_2 are convex functions on $\mathbf{R}$, so quasiconvex on $\mathbf{R}$, but f is not a quasiconvex function on $\mathbf{R}$ as for $x = 1$, $u = 3$, $\lambda = 1/2$,

$$f(\lambda x + (1-\lambda)u) = f(2) = 0 > Max\{f(x), f(u)\} = -1.$$

The next result is important as it completely characterizes the class of quasiconvex functions.

Theorem 12.2.1 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. The level sets of f are given by $\Gamma_\alpha = \{x \in S : f(x) \leq \alpha\}$, $\alpha \in \mathbf{R}$. f is a quasiconvex function on S if and only if Γ_α is a convex set, for every $\alpha \in \mathbf{R}$.*

Proof. Assume that f is a quasiconvex function on S. If for $\alpha \in \mathbf{R}$, Γ_α is either an empty set or a singleton set then the result follows trivially. Suppose $x, u \in \Gamma_\alpha$. Then

$$f(x) \leq \alpha, \ f(u) \leq \alpha,$$

which in view of quasiconvexity yields

$$f(\lambda x + (1-\lambda)u) \leq Max\{f(x), f(u)\} \leq \alpha, \quad \forall\, \lambda \in [0,1].$$

Consequently, $\lambda x + (1-\lambda)u \in \Gamma_\alpha$, $\forall\, \lambda \in [0,1]$, implying Γ_α is a convex set.

For converse, let $x, u \in S$ be such that $f(x) \leq f(u)$. Then, $x, u \in \Gamma_{f(u)}$. By virtue of convexity of the set Γ_α, for every $\alpha \in \mathbf{R}$, we get that, $\lambda x + (1-\lambda)u \in \Gamma_{f(u)}$, $\forall\, \lambda \in [0,1]$. Thereby giving

$$f(x) \leq f(u) \Rightarrow f(\lambda x + (1-\lambda)u) \leq f(u), \quad \forall\, \lambda \in [0,1].$$

Thus, f is a quasiconvex function on S. □

Analogous result can be worked out for quasiconcave functions in terms of their upper level sets.

Corollary 12.2.1 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. The upper level sets of f are given by $\Omega_\alpha = \{x \in S : f(x) \geq \alpha\}$, $\alpha \in \mathbf{R}$. f is a quasiconcave function on S if and only if Ω_α is a convex set, for every $\alpha \in \mathbf{R}$.*

The above characterization of quasiconvex (respectively, quasiconcave) function is useful and can serve as an alternate definition of quasiconvex (respectively, quasiconcave) function.

Going back to the example, $f(x) = [x]$, $x \in \mathbf{R}$, the greatest integer function, we find that f is neither a convex function nor a concave function on $\mathbf{R}$ but f is both quasiconvex and quasiconcave function on $\mathbf{R}$. It can easily be verified that for any integer x, and $x \leq \alpha < x+1$, we have $\Gamma_\alpha = (-\infty, x+1)$ and $\Omega_\alpha = [x, \infty)$. Thus, quasiconvexity (respectively, quasiconcavity) is a genuine generalization of convexity (respectively, concavity).

Geometrically, Theorem 12.2.1 can be interpreted as follows. A function f is quasiconvex if and only if the line segment joining any two points on any two level curves of f lies nowhere above the level curve of f corresponding to the higher value of f.

In the first figure in Fig 12.1, a point $\hat{x}$ lies on the level curve $f(x) = \alpha_1$ and a point u lies on the level curve $f(x) = \alpha_2$, i.e. $f(\hat{x}) = \alpha_1$ and $f(u) = \alpha_2$. Here, α_1, α_2 are any

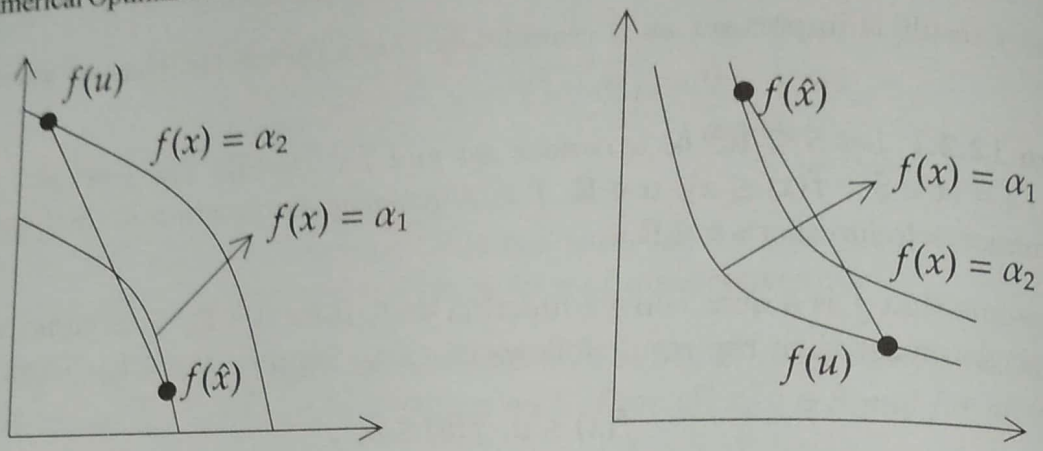


Fig. 12.1.

two real constants. Then any point on the line segment joining $\hat{x}$ and u is of the form $\lambda\hat{x} + (1-\lambda)u$, and one can see that $f(\lambda\hat{x} + (1-\lambda)u) \leq \alpha_2,\ \forall\, \lambda \in [0,1]$.

Similar interpretation can be used to describe Corollary 12.2.1, i.e. a function f is quasiconcave if and only if the line segment joining any two points on any two upper level curves of f lies nowhere below the upper level curve of f corresponding to the minimum value of f. This statement is depicted in the second figure in Fig 12.1.

At this point, one may wonder as to why so much of discussion is centered around the level sets? We pause here to answer this question from economic point of view. In fact one of the earliest applications of quasiconvexity/quasiconcavity appeared in economic theory.

The standard model of decision making in economic theory consists of a set of alternatives and an appropriate preference relation over these alternatives. To facilitate the analysis, the preference relation is assumed to be a complete, reflexive and transitive relation. This relation is generally described by a continuous function called the 'utility function', denoted by U. Here, by continuity of the utility function, we mean that a small change in the consumption bundle (which mathematically is represented by a vector in $\mathbf{R}^n$) causes only a small change in the preference (or utility) level. The decision maker is supposed to choose his most preferred alternative with the property that it has the highest utility attached with it.

For example, if $U(x) = 6$ and $U(y) = 3$ then the bundle x is strictly preferred to the bundle y. But in no way we mean that x is two times better than y. Consider the bundles $x_1 = (4,1)$, $x_2 = (2,3)$ and $x_3 = (2,2)$ with $U(x_1) = 4$, $U(x_2) = 6$ and $U(x_3) = 4$. Then, x_2 is strictly preferred over x_1 and x_3, while x_1 and x_3 are at the same preference level. Here, we would like to point out that the numbers assigned to the bundles are not important so far as their preference order is maintained. Thus, the interpretation would remain unaltered if we take another utility function, say U_1, with $U_1(x_1) = 18$, $U_1(x_2) = 20$ and $U_1(x_3) = 18$, in the above case. The bundles at the same utility level form a curve which is called an 'indifference curve' in economics. In the above example, bundles $(4,1)$ and

(2,2) are on the indifference curve with utility level $U = 4$, but the bundle (2,3) is on the indifference curve with utility level $U = 6$.

The collection of all the indifference curves, each with its assigned utility level, completely represent the consumers' preferences. It is generally observed with most commonly used utility functions that the upper level sets of the indifference curves are convex sets. This is equivalent to the condition that the utility function representing the consumer preference is a quasiconcave function. Therefore, in economic models, the critical issue about the function is not whether it is convex or concave but simply whether its upper or lower level sets are convex sets.

Returning back to our earlier discussion, we present some more properties of quasiconvex and quasiconcave functions.

Theorem 12.2.2 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be a differentiable quasiconvex function on S. Then*

$$x, u \in S,\ f(x) \leq f(u) \Rightarrow \nabla f(u)^T(x-u) \leq 0.$$

Proof. If $x = u$ then the implication holds trivially. Suppose $x \neq u$. Since S is an open set, we can find a ball around u, say $N_\delta(u)$, such that $N_\delta(u) \subset S$. Here, $\delta > 0$ can be chosen sufficiently small so that $x \notin N_\delta(u)$. Let $\bar{\lambda} \in \left(0, \dfrac{\delta}{\|x-u\|}\right)$, with $\bar{\lambda} < 1$, and define $\bar{x} = \bar{\lambda}x + (1-\bar{\lambda})u$. Then, $\bar{x} \in N_\delta(u)$. Quasiconvexity of f at u yields

$$f(\bar{x}) = f(\bar{\lambda}x + (1-\bar{\lambda})u) \leq f(u).$$

Thus,
$$f(u + \bar{\lambda}(x-u)) - f(u) \leq 0.$$

Using differentiability of f at u, we get,

$$\nabla f(u)^T(x-u) + \|x-u\|\beta(u, \bar{\lambda}(x-u)) \leq 0,$$

where $\beta : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ is such that

$$\lim_{w \to 0} \frac{\beta(u,w)}{\|w\|} = 0.$$

Now, taking limit as $\bar{\lambda} \to 0$, we get

$$\nabla f(u)^T(x-u) \leq 0. \qquad \square$$

Corollary 12.2.2 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be a differentiable quasiconcave function on S. Then*

$$x, u \in S,\ f(x) \geq f(u) \Rightarrow \nabla f(u)^T(x-u) \geq 0.$$

Recall from Chapter 7 that a twice differentiable function $f : S \to \mathbf{R}$, on an open convex set $S \subseteq \mathbf{R}^n$, is convex on S if its Hessian matrix $H_f(x)$ is a positive semidefinite matrix. Parallel to this result, we have a second order characterization of quasiconvex function. For this, we need to understand the concept of *bordered Hessian*, $B_f(x)$, of f defined as follows

$$B_f(x) = \begin{pmatrix} 0 & \nabla f(x)^T \\ \nabla f(x) & H_f(x) \end{pmatrix}.$$

$B_f(x)$ is named bordered Hessian matrix as the Hessian matrix $H_f(x)$ of f is bordered by an additional row and column from the top and the left, respectively. Thus, $B_f(x)$ is a matrix of the order $(n+1) \times (n+1)$. The principal minors of $B_f(x)$ are given by

$$D_k(x) = \begin{vmatrix} 0 & f_1'(x) & f_2'(x) & \dots & f_k'(x) \\ f_1'(x) & f_{11}''(x) & f_{12}''(x) & \dots & f_{1k}''(x) \\ f_2'(x) & f_{21}''(x) & f_{22}''(x) & \dots & f_{2k}''(x) \\ \vdots & \vdots & \vdots & & \vdots \\ f_k'(x) & f_{k1}''(x) & f_{k2}''(x) & \dots & f_{kk}''(x) \end{vmatrix}, \quad (k = 1, \dots, n).$$

Here by $f_{ij}''(x)$ we mean $\frac{\partial^2 f(x)}{\partial x_i \partial x_j}$. Note that for a function f of n variables there are n principal minors, $D_1(x), \dots, D_n(x)$, of $B_f(x)$.

For example, consider a function $f : \mathbf{R}^2 \to \mathbf{R}$ defined as $f(x, y) = ye^{-x}$. Then the principal minors of its bordered Hessian matrix $B_f(x)$ are

$$D_1(x, y) = \begin{vmatrix} 0 & f_1' \\ f_1'(x, y) & f_{11}''(x, y) \end{vmatrix} = \begin{vmatrix} 0 & -ye^{-x} \\ -ye^{-x} & ye^{-x} \end{vmatrix} = -y^2 e^{-2x}.$$

$$D_2(x, y) = \begin{vmatrix} 0 & f_1' & f_2'(x, y) \\ f_1'(x, y) & f_{11}''(x, y) & f_{12}''(x, y) \\ f_2'(x, y) & f_{21}''(x, y) & f_{22}''(x, y) \end{vmatrix} = \begin{vmatrix} 0 & -ye^{-x} & e^{-x} \\ -ye^{-x} & ye^{-x} & -e^{-x} \\ e^{-x} & -e^{-x} & 0 \end{vmatrix} = ye^{-3x}.$$

We state below a characterization of a twice differentiable quasiconvex function without proof.

Theorem 12.2.3 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$ be a twice differentiable function. A necessary condition for f to be a quasiconvex function on S is that $D_k(x) \leq 0$ $(k = 1, \dots, n)$, $\forall\, x \in S$, whereas, a sufficient condition for f to be a quasiconvex function on S is that $D_k(x) < 0$ $(k = 1, \dots, n)$, $\forall\, x \in S$.*

In the previous example, $f(x, y) = ye^{-x}$, we find that $D_1(x, y) \leq 0, \forall\, (x, y) \in \mathbf{R}^2$, while $D_2(x, y) \leq 0$, $\forall\, (x, y) \in \mathbf{R}^2$ with $y \leq 0$. Thus, $f(x, y) = ye^{-x}$ is a quasiconvex function on $\mathbf{R} \times \mathbf{R}_-$. One can easily verify that f is not a convex function on $\mathbf{R} \times \mathbf{R}_-$.

Corollary 12.2.3 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$ be a twice differentiable function. A necessary condition for f to be a quasiconcave function on S is that $D_1(x) \leq 0, D_2(x) \geq 0, \ldots, D_n(x) \leq 0$, if n is odd, and $D_n(x) \geq 0$ if n is even, $\forall x \in S$, whereas, a sufficient condition for f to be a quasiconcave function on S is that $D_1(x) < 0, D_2(x) > 0, \ldots, D_n(x) < 0$, if n is odd, and $D_n(x) > 0$ if n is even, $\forall x \in S$.*

For example, consider a function $f(x, y) = xy$, on the set $S = \{(x, y) : x > 0,\ y > 0\}$. Then,

$$D_1(x,y) = \begin{vmatrix} 0 & f_1' \\ f_1'(x,y) & f_{11}''(x,y) \end{vmatrix} = \begin{vmatrix} 0 & y \\ y & 0 \end{vmatrix} = -y^2 < 0.$$

$$D_2(x,y) = \begin{vmatrix} 0 & f_1' & f_2'(x,y) \\ f_1'(x,y) & f_{11}''(x,y) & f_{12}''(x,y) \\ f_2'(x,y) & f_{21}''(x,y) & f_{22}''(x,y) \end{vmatrix} = \begin{vmatrix} 0 & y & x \\ y & 0 & 1 \\ x & 1 & 0 \end{vmatrix} = 2xy > 0 \text{ on } S.$$

Hence, f is a quasiconcave function on S. Observe that f is not concave on S as

$$f\left(\frac{1}{2}(0,0) + \frac{1}{2}(2,2)\right) < \frac{1}{2}f(0,0) + \frac{1}{2}f(2,2).$$

Here, we may also note that the level curves of f, $\{(x, y) \in S : f(x, y) = \alpha\}$, $\alpha \in \mathbf{R}$, are rectangular hyperbolae. So, the upper level sets of f, $\{(x, y) \in S : f(x, y) \geq \alpha\}$, are convex sets, thereby, confirming the quasiconcavity of f on S.

Theorem 12.2.4 *Let $S \subseteq \mathbf{R}^n$ be a polytope and $f : S \to \mathbf{R}$ be a continuous function on S. Then f possesses a maxima (respectively, minima) at an extreme point of S and in all its polyhedral subsets if and only if f is a quasiconvex (respectively, quasiconcave) function on S.*

Proof. The proof of quasiconvex case is given. The quasiconcave case can be discussed in a similar manner.

Let f be a quasiconvex function on S. Since S is a polytope, it possesses finite number of extreme points. Suppose $x_1, \ldots, x_p$ are the extreme points of S such that $f(x_j) = \underset{1 \leq i \leq p}{Max} f(x_i)$. Let $x \in intS$. Then there exist $\lambda_1, \ldots, \lambda_p$ such that

$$x = \sum_{i=1}^{p} \lambda_i x_i, \quad \lambda_i \geq 0 \ \ (i = 1, \ldots, p), \quad \sum_{i=1}^{p} \lambda_i = 1.$$

By successive application of the definition of quasiconvexity of f, we get

$$f(x) = f\left(\sum_{i=1}^{p} \lambda_i x_i\right) \leq f(x_j) = \underset{1 \leq i \leq p}{Max} f(x_i).$$

Thus, the maxima of f is attained at an extreme point of S.

Conversely, suppose f possesses a maxima at an extreme point of S and all its polyhedral subsets. Then, in particular, a line segment joining two points $x, u \in S$ is a bounded polyhedral subset of S, so, f possesses its maxima at an extreme point which, in this case, is either x or u. Thus,

$$f(\lambda x + (1-\lambda)u) \leq Max\{f(x), f(u)\}, \ \forall \lambda \in [0,1].$$

Consequently, f is a quasiconvex function on S.

We now illustrate the above theorem by an example. □

Example 12.2.1 *Obtain the optimal solution of the following nonlinear optimization problem*

$$\begin{aligned} &Max && f(x,y) = x^3 + xy \\ &\text{subject to} \\ & && x - y \geq 1 \\ & && x + 3y \geq -3 \\ & && 2x \leq 1. \end{aligned}$$

Solution The feasible set S is a convex polytope with extreme points $(0,-1)$, $\left(\frac{1}{2}, \frac{-7}{6}\right)$ and $\left(\frac{1}{2}, \frac{-1}{2}\right)$. Also, f is a quasiconvex function on S because

$$D_1(x,y) = \begin{vmatrix} 0 & f_1' \\ f_1'(x,y) & f_{11}''(x,y) \end{vmatrix} = \begin{vmatrix} 0 & 3x^2+y \\ 3x^2+y & 6x \end{vmatrix} = -(3x^2+y)^2 < 0 \text{ on } S.$$

$$D_2(x,y) = \begin{vmatrix} 0 & f_1' & f_2'(x,y) \\ f_1'(x,y) & f_{11}''(x,y) & f_{12}''(x,y) \\ f_2'(x,y) & f_{21}''(x,y) & f_{22}''(x,y) \end{vmatrix} = \begin{vmatrix} 0 & 3x^2+y & x \\ 3x^2+y & 6x & 1 \\ x & 1 & 0 \end{vmatrix}$$

$$= 2xy < 0 \text{ on } S.$$

Next, note that for any $(x,y) \in S$,

$$0 \leq x \leq \frac{1}{2} \text{ and } -\frac{7}{6} \leq y \leq -\frac{1}{2} \Rightarrow f(x,y) = x^3 + xy \leq x^3 - \frac{1}{2}x \leq 0 = f(0,-1),$$

and hence the optimal solution is $x^\star = (0,-1)$ which is an extreme point of S.

We next present a result that will be used in the subsequent section on fractional programming problems.

Theorem 12.2.5 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f, g : S \to \mathbf{R}$, $g(x) \neq 0$, $\forall x \in S$. Define a function $\theta : S \to \mathbf{R}$ as $\theta(x) = \dfrac{f(x)}{g(x)}$, $x \in S$. Then, θ is a quasiconvex function on S if any of the following condition is satisfied*

(i) *f is convex on S, g is linear on S,* $g(x) > 0, \forall x \in S$.
(ii) *f is concave on S, g is linear on S,* $g(x) < 0, \forall x \in S$.
(iii) *f and g are convex on S,* $f(x) \leq 0$, $g(x) > 0, \forall x \in S$.
(iv) *f and g are concave on S,* $f(x) \geq 0$, $g(x) < 0, \forall x \in S$.
(v) *f is concave on S, g is convex on S,* $f(x) \leq 0$, $g(x) < 0, \forall x \in S$.
(vi) *f is convex on S, g is concave on S,* $f(x) \geq 0$, $g(x) > 0, \forall x \in S$.

Proof. We shall be proving the theorem for the case (*i*), case (*iii*) and case (*v*). The rest of the cases follow on the similar lines.

Consider the level sets of θ as

$$\Gamma_\alpha = \{x \in S : \theta(x) \leq \alpha\}, \ \alpha \in \mathbf{R}.$$

Suppose condition (*i*) is true. Then,

$$\Gamma_\alpha = \{x \in S : f(x) - \alpha g(x) \leq 0\}, \ \alpha \in \mathbf{R}.$$

As f is a convex function and g is a linear function, so, for any $\alpha \in \mathbf{R}$, $f - \alpha g$ is a convex function on S. Thus, Γ_α is the zero-level set of the convex function, $f - \alpha g$, and hence a convex set. The level set characterization of quasiconvexity (Theorem 12.2.1) implies that θ is a quasiconvex function on S.

If case (*iii*) holds then

$$\Gamma_\alpha = \{x \in S : f(x) - \alpha g(x) \leq 0\}, \ \alpha \in \mathbf{R}.$$

For $\alpha \leq 0$, $f - \alpha g$ is a convex function on S.

Let $\alpha > 0$. By assumption,

$$\frac{f(x)}{g(x)} \leq 0, \ \forall x \in S \Rightarrow \frac{f(x)}{g(x)} \leq \alpha, \ \forall x \in S.$$

Thus, for $\alpha \leq 0$, the level set Γ_α is a zero-level set of a convex function $f - \alpha g$ and for $\alpha > 0$, Γ_α is the whole of S. In both cases, Γ_α is a convex set. Thus, θ is a quasiconvex function on S.

In case (*v*),

$$\Gamma_\alpha = \{x \in S : f(x) - \alpha g(x) \geq 0\}, \ \alpha \in \mathbf{R}.$$

If $\alpha \geq 0$ then $f - \alpha g$ is a concave function. So, Γ_α is a zero-upper level set of a concave function, hence a convex set. Let $\alpha < 0$. By assumption,

$$\frac{f(x)}{g(x)} \geq 0, \ \forall x \in S.$$

Thus, $\Gamma_\alpha = \emptyset$, hence a convex set by convention.

In both cases (i.e. $\alpha \geq 0$, $\alpha < 0$), Γ_α is a convex set making θ a quasiconvex function on S. □

Analogous result can be stated and proved for quasiconcave function.

Recall that a function $f : S \to \mathbf{R}$, where $S \subseteq \mathbf{R}^n$ is a convex set, is called linear function on S if and only if it is both convex and concave function on S. Thus $f(x) = c^T x + \alpha$, $x, c \in \mathbf{R}^n$, $\alpha \in \mathbf{R}$, is a linear function on $\mathbf{R}^n$. From the above discussion, we observe that the ratio of two linear functions defined by

$$\theta(x) = \frac{c^T x + \alpha}{d^T x + \beta}, \quad x \in S \subseteq \mathbf{R}^n, \ c, d \in \mathbf{R}^n, \ \alpha, \beta \in \mathbf{R},$$

with $d^T x + \beta > 0$, $\forall x \in S$ or $d^T x + \beta < 0$, $\forall x \in S$, is both a quasiconvex and a quasiconcave function on S. We shall be needing this result in the later part of this chapter when we shall be studying linear fractional programming problems.

We like to point out here that a function f of the type $f(x) = c^T x + \alpha$, $x \in S \subseteq \mathbf{R}^n, c \in \mathbf{R}^n, \alpha \in \mathbf{R}$, is sometimes termed as an affine function in literature. This is because in linear algebra, linearity of the function is described by the relation

$$f(\mu x + \delta u) = \mu f(x) + \delta f(u), \ \forall \mu, \delta \in \mathbf{R}, \ \forall x, u \in S.$$

Obviously for the linear function f, in the sense of this equation, $f(0) = 0$. But for the function $f(x) = c^T x + \alpha$, $f(0) \neq 0$, unless $\alpha = 0$. So, in order to distinguish between functions $x \mapsto c^T x$ and $x \mapsto c^T x + \alpha$, the former function is called a linear function and the latter one is called an affine function. But in this text, without any ambiguity, we have been calling, $f(x) = c^T x + \alpha$, a linear function. It is to be understood in the sense of optimization that it is both convex and concave function. In the same spirit we define the following concept.

Definition 12.2.3 (Quasilinear Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a quasilinear function on S if it is both quasiconvex and quasiconcave function on S.*

For instance, $f(x) = x^3$, $x \in \mathbf{R}$, or the ratio of two linear functions θ described above are quasilinear functions.

Undoubtably the most powerful property of convex function is that every local min point of convex function is its global min point. The entire nonlinear convex optimization rely on this fact. In the discussion to follow we try to explore whether such a property exists for a class of quasiconvex functions.

Theorem 12.2.6 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$ be a quasiconvex function on S. Then every strict local min point of f is a strict global min point of f on S.*

Proof. Let $u \in S$ be a strict local min point of f that is not a strict global min point of f. Then there exist $\delta > 0$ and $y \in S$ such that

$$f(u) < f(x),\ \forall\, x \in N_\delta(u) \cap S \text{ and } f(y) \le f(u).$$

By quasiconvexity of f,

$$f(\lambda y + (1-\lambda)u) \le f(u),\ \forall\, \lambda \in [0,1]. \tag{12.1}$$

Also, $y \notin N_\delta(u)$. Choose $\bar{\lambda} \in \left(0, \dfrac{\delta}{\|y-u\|}\right)$ with $\bar{\lambda} < 1$. Then, $\bar{\lambda}y + (1-\bar{\lambda})u \in N_\delta(u) \cap S$. Hence, we obtain

$$f(u) < f(\bar{\lambda}y + (1-\bar{\lambda})u). \tag{12.2}$$

(12.1) and (12.2) are not compatible with each other. Consequently, u is a strict global min point of f on S. □

Corollary 12.2.4 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$ be a quasiconcave function on S. Then every strict local max point of f is a strict global max point of f on S.*

The above results fail if the phrase 'strict' is dropped. In fact a local min point (respectively, local max point) of a quasiconvex (respectively, quasiconcave) function is not necessarily its point of global min (respectively, global max). Fig 12.2 provides a graphical illustration of this statement.

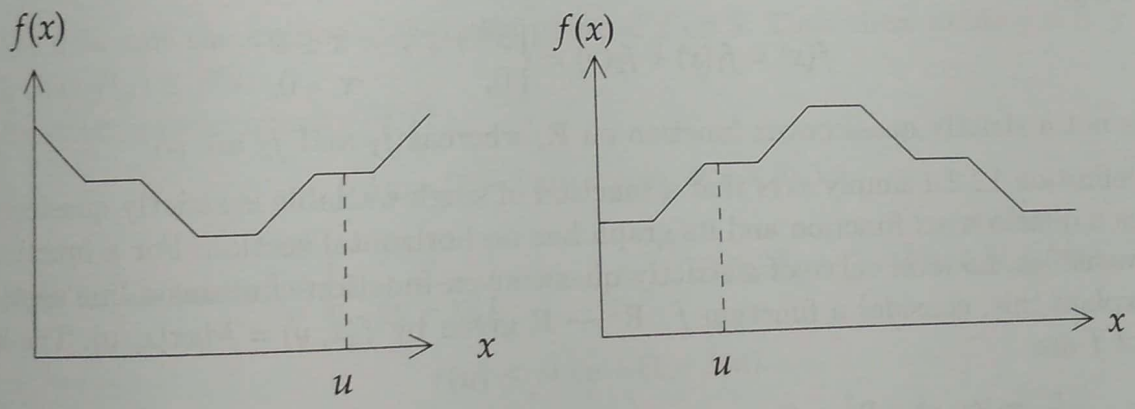


Fig. 12.2.

The upside down podium like function in the first figure in Fig 12.2 is a quasiconvex function as its level sets are convex intervals. Here, point u is a local min point of f but it is not a global min point of f. In the second figure in Fig 12.2, the function is a quasiconcave function as its upper level sets are convex sets. Also, u is a local max point of the function but obviously it is not a global max point of the function.

The above example(s) indicates the need to introduce a stricter version of quasiconvexity (respectively, quasiconcavity) called strictly quasiconvex (respectively, quasiconcave) function that can possess the local-global extremum property.

Definition 12.2.4 (Strictly Quasiconvex Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a strictly quasiconvex function on S if*

$$x, u \in S,\ x \neq u,\ f(x) \leq f(u) \Rightarrow f(\lambda x + (1-\lambda)u) < f(u), \quad \forall\ \lambda \in (0,1).$$

Definition 12.2.5 (Strictly Quasiconcave Function). *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$. Then f is called a strictly quasiconcave function on S if*

$$x, u \in S,\ x \neq u,\ f(x) \geq f(u) \Rightarrow f(\lambda x + (1-\lambda)u) > f(u), \quad \forall\ \lambda \in (0,1).$$

From the above definitions, we can easily observe the following

(i) f is strictly quasiconcave on S if and only if $-f$ is strictly quasiconvex on S.

(ii) A strictly quasiconvex (respectively, quasiconcave) function is always a quasiconvex function on S but the converse, in general, is not true. For example, $f(x) = [x]$, $x \in \mathbf{R}$, the greatest integer function, is both quasiconvex and quasiconcave function on $\mathbf{R}$ but neither strictly quasiconvex nor strictly quasiconcave function on $\mathbf{R}$.

(iii) Sum of two strictly quasiconvex (respectively, quasiconcave) functions need not be strictly quasiconvex (respectively, quasiconcave) function on S. To see this, let $f_1, f_2 : \mathbf{R} \to \mathbf{R}$ be defined as

$$f_1(x) = \begin{cases} x^3, & x \leq 0 \\ x, & x > 0; \end{cases} \qquad f_2(x) = \begin{cases} x^2, & x \leq 0 \\ -x, & x > 0. \end{cases}$$

Then,

$$f(x) = f_1(x) + f_2(x) = \begin{cases} x^3 + x^2, & x \leq 0 \\ 0, & x > 0, \end{cases}$$

is not a strictly quasiconvex function on $\mathbf{R}$, whereas f_1 and f_2 are so.

Definition 12.2.4 simply says that a function of single variable is strictly quasiconvex if it is a quasiconvex function and its graph has no horizontal section. For a function of two variables, no level curve of a strictly quasiconvex function contains a line segment. To explain this, consider a function $f : \mathbf{R}^2 \to \mathbf{R}$ given by $f(x, y) = Max\{x, y\}$. The level sets of f are

$$\Gamma_\alpha = \{(x, y) \in \mathbf{R}^2 : f(x, y) \leq \alpha\} = \{(x, y) \in \mathbf{R}^2 : x \leq \alpha,\ y \leq \alpha\},\ \alpha \in \mathbf{R}.$$

All Γ_α are upside down right angled with vertices on a ray from origin (see, Fig 12.3). f is a quasiconvex function on $\mathbf{R}^2$ but it is not a strictly quasiconvex function as for $x = (1,0)$, $u = (1,1)$, $x \neq u$, $f(x) < f(u)$, but $f\left(\frac{1}{2}x + \frac{1}{2}u\right) > f(u)$.

Observe that the level curves of f contain line segments.

We would like to caution the readers that in certain texts, function satisfying Definition 12.2.4 (respectively, Definition 12.2.5) is called strongly quasiconvex (respectively, quasiconcave) function.

We now present a result that had motivated us to introduce the notion of strict quasiconvexity (respectively, quasiconcavity).

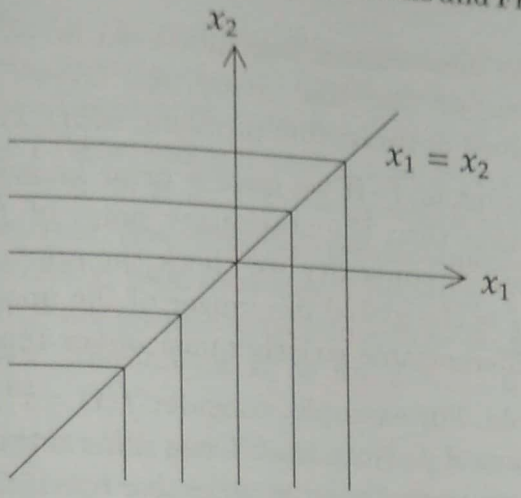


Fig. 12.3.

Theorem 12.2.7 *Let $S \subseteq \mathbf{R}^n$ be a convex set and $f : S \to \mathbf{R}$ be a strictly quasiconvex (respectively, strictly quasiconcave) function on S. Then every local min point (respectively, local max point) of f is its unique global min point (respectively, global max point) on S.*

Proof. Let $u \in S$ be a local min point of f. Then there exists $\delta > 0$ such that

$$f(u) \leq f(x), \ \forall\, x \in N_\delta(u) \cap S.$$

Let u be not the unique global min point of f on S. Then there exists $y \in S$, $y \neq u$, such that $f(y) \leq f(u)$.

By strict quasiconvexity of f,

$$f(\lambda y + (1-\lambda)u) < f(u), \ \forall\, \lambda \in (0,1). \tag{12.3}$$

Now, we choose a $\bar{\lambda} \in \left(0, \dfrac{\delta}{\|y - x\|}\right)$ with $\bar{\lambda} < 1$. Then, $\bar{\lambda} y + (1-\bar{\lambda})u \in N_\delta(u) \cap S$. Thus,

$$f(u) \leq f(\bar{\lambda} y + (1-\bar{\lambda})u).$$

But this contradicts (12.3). Therefore, we must have $f(u) < f(x)$, $\forall\, x \in S$, yielding the required result. □

This property of strictly quasiconvex (respectively, quasiconcave) function is very useful particularly for the unconstrained minimization (respectively, maximization) algorithms which do not impose any differentiability condition on the objective function. In fact strict quasiconvexity (respectively, quasiconcavity) of the function ensures its unimodularity in the interval of uncertainty within which the optimal solution for the objective function is searched, like in Fibonacci search technique or golden section method.

It is important to note that quasiconvexity or quasiconcavity and their strict generalizations do not require the function to be differentiable in the domain. In many

cases, functions belonging to these classes may not even be continuous. Looking from the algorithmic point of view, we find that many well known numerical optimization techniques for the unconstrained optimization problem, $\underset{x \in \mathbf{R}^n}{Min} f(x)$, are gradient based and search only the stationary point of f. If we have a prior knowledge that the objective function f is a convex function then the stationary point of f is definitely its global minimizer. However, if instead of convexity we check the applicability of this property (i.e. the stationary point of f is the global minimizer of the unconstrained optimization problem, $\underset{x \in \mathbf{R}^n}{Min} f(x)$) for the differentiable strictly quasiconvex function, then we find that the same assertion fails to hold. For example, consider $f(x) = x^3$, $x \in \mathbf{R}$. Then $\nabla f(0) = 0$ but $x = 0$ is a point of inflexion of f. Note that f is a differentiable strictly quasiconvex function on $\mathbf{R}$. Thus, even if we know that the objective function f in an unconstrained minimization problem is differentiable strictly quasiconvex, the gradient based search technique will only provide a stationary point of f but the nature of the stationary point can not be judged from the information of strict quasiconvexity.

These observations lead us to introduce two new classes of functions that share an important property with convex functions that the stationary points are the global min points. We describe these functions in the next section.

12.3 Pseudoconvex and Pseudoconcave Functions

Definition 12.3.1 (Pseudoconvex Function). *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$. Then f is called a pseudoconvex function on S if it is differentiable and*

$$x, u \in S,\ \nabla f(u)^T(x-u) \geq 0 \Rightarrow f(x) \geq f(u).$$

Equivalently,

$$x, u \in S,\ f(x) < f(u) \Rightarrow \nabla f(u)^T(x-u) < 0.$$

Definition 12.3.2 (Pseudoconcave Function). *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$. Then f is called a pseudoconcave function on S if it is differentiable and*

$$x, u \in S,\ \nabla f(u)^T(x-u) \leq 0 \Rightarrow f(x) \leq f(u).$$

Equivalently,

$$x, u \in S,\ f(x) > f(u) \Rightarrow \nabla f(u)^T(x-u) > 0.$$

From the above two definitions, we observe the following

(i) f is a pseudoconcave function on S if and only if $-f$ is a pseudoconvex function on S.

(ii) If f is a differentiable convex (respectively, concave) function on S then f is a pseudoconvex (respectively, pseudoconcave) function on S. The reverse of this statement is not necessarily true. To see this, consider $f(x) = x^3 + x$, $x \in \mathbf{R}$. Then f is a pseudoconvex function as $f'(u)(x-u) \geq 0 \Rightarrow x \geq u \Rightarrow x^3 + x \geq u^3 + u \Rightarrow f(x) \geq f(u)$, but f is not a convex function on $\mathbf{R}$ as $f''(x) = 6x$ which is not positive (definite) on $\mathbf{R}$.

(iii) Sum of two pseudoconvex (respectively, pseudoconcave) functions need not be a pseudoconvex (respectively, pseudoconcave) function. Take $f_1(x) = -x$, $f_2(x) = x^3 + x$, $x \in \mathbf{R}$. Then f_1 and f_2 are pseudoconvex functions on $\mathbf{R}$ but $f(x) = f_1(x) + f_2(x) = x^3$ is not a pseudoconvex function on $\mathbf{R}$ as for $x = 1$, $u = 0$, $f'(u)(x-u) = 0$ but $f(x) > f(u)$.

(iv) (a) Every stationary point of a pseudoconvex (respectively, pseudoconcave) function is its global min point (respectively, max point). In means that if f is a pseudoconvex (respectively, pseudoconcave) function at $u \in S$ and $\nabla f(u) = 0$ then u is global min point (respectively, max point) of f on S. For example, consider a function $f : \mathbf{R} \to \mathbf{R}$ defined by

$$f(x) = \begin{cases} (x-1)^2, & x \leq 2 \\ 2x - 3, & x > 2, \end{cases}$$

then f is a differentiable function with

$$f'(x) = \begin{cases} 2(x-1), & x \leq 2 \\ 2, & x > 2. \end{cases}$$

By taking three different cases, viz., $u \geq 2$; $1 < u < 2$; $u \leq 1$, it can easily be verified that $f'(u)(x-u) \geq 0 \Rightarrow f(x) \geq f(u)$, $x \in \mathbf{R}$. Thus, f is a pseudoconvex function on $\mathbf{R}$. Also, $x = 1$ is the only stationary point of f which is a global min point of f on $\mathbf{R}$.

(iv) (b) As a consequence of (iv)(a), we can assert that if a differentiable function f possesses a stationary point $u \in S$ which is not a global min point (respectively, global max point) of f then f can not be a pseudoconvex (respectively, pseudoconcave) function. To justify it, let $f(x) = \sin x$, $-\pi \leq x \leq \pi$. f is not a pseudoconvex function on $[-\pi, \pi]$ as, $f'(\frac{\pi}{2})(x - \frac{\pi}{2}) = 0$ for $x = 0$ but $f(x) < f(\frac{\pi}{2})$. Here, take a note that f has two stationary points $u_1 = -\frac{\pi}{2}$ and $u_2 = \frac{\pi}{2}$ but u_2 is not the global min point of f.

(v) Geometrically, pseudoconvexity (respectively, pseudoconcavity) of a function f can be interpreted as follows. If the directional derivative of f at any point u in the direction $(x-u)$ is nonnegative (respectively, nonpositive) then the values of f are nondecreasing (respectively, nonincreasing) in that direction.

The last observation along with (iv)(a) are significant from the computational view point.

The next result pertains to the conditions for the ratio of two functions to be a pseudoconvex function.

Theorem 12.3.1 *Let $S \subseteq \mathbf{R}^n$ be an open set and $f, g : S \to \mathbf{R}$ be differentiable functions at $u \in S$ with $g(x) \neq 0$, $\forall x \in S$. Then, a function $\theta : S \to \mathbf{R}$ defined as $\theta(x) = \dfrac{f(x)}{g(x)}$, $x \in S$, is a pseudoconvex function at $u \in S$ if any of the following conditions hold*

(i) f is convex at u, g is linear on S, $g(x) > 0$, $\forall x \in S$.
(ii) f is concave at u, g is linear on S, $g(x) < 0$, $\forall x \in S$.
(iii) f and g are convex at u, $f(x) \leq 0$, $g(x) > 0$, $\forall x \in S$.
(iv) f and g are concave at u, $f(x) \geq 0$, $g(x) < 0$, $\forall x \in S$.
(v) f is concave at u, g is convex at u, $f(x) \leq 0$, $g(x) < 0$, $\forall x \in S$.
(vi) f is convex at u, g is concave at u, $f(x) \geq 0$, $g(x) > 0$, $\forall x \in S$.

Proof. We shall be proving only the case (*iii*) and case (*v*) of the above stated conditions. The rest of the cases can be discussed using the same ideas.

Suppose case (*iii*) holds. Let $x, u \in S$. Using convexity of f and g at u, we get

$$f(x) - f(u) \geq \nabla f(u)^T (x - u)$$
$$g(x) - g(u) \geq \nabla g(u)^T (x - u).$$

Multiplying the first inequality by $g(u) > 0$ and the second inequality by $f(u) \leq 0$, and subtracting

$$\begin{aligned} f(x)g(u) - f(u)g(x) &\geq (g(u)\nabla f(u) - f(u)\nabla g(u))^T (x - u) \\ &= (g(u))^2 (\nabla \frac{f}{g}(u))^T (x - u) \\ &= (g(u))^2 \nabla \theta(u)^T (x - u). \end{aligned}$$

Now, suppose

$$\begin{aligned} \nabla\theta(u)^T (x - u) \geq 0 &\Rightarrow (g(u))^2 \nabla\theta(u)^T (x - u) \geq 0 \\ &\Rightarrow f(x)g(u) \geq f(u)g(x) \\ &\Rightarrow \frac{f(x)}{g(x)} \geq \frac{f(u)}{g(u)} \\ &\Rightarrow \theta(x) \geq \theta(u). \end{aligned}$$

Thereby yielding that θ is a pseudoconvex function o S.

Suppose condition (*v*) holds. Let $x, u \in S$. Since f is a concave function at u and g is a convex function at u, so,

$$f(x) - f(u) \leq \nabla f(u)^T (x - u)$$
$$g(x) - g(u) \geq \nabla g(u)^T (x - u).$$

Multiplying the first inequality by $g(u) < 0$ and the second inequality by $f(u) \leq 0$, and subtracting, we get

$$\begin{aligned} f(x)g(u) - f(u)g(x) &\geq (g(u)\nabla f(u) - f(u)\nabla g(u))^T (x - u) \\ &= (g(u))^2 \nabla\theta(u)^T (x - u). \end{aligned}$$

The rest of the arguments are the same as in the latter part of the case (*iii*) discussed above.

Analogous results can be stated and proved for pseudoconcave function. □

From the above discussion it is clear that a function $\theta(x) = \dfrac{c^T x + \alpha}{d^T x + \beta}$, $x \in S \subseteq \mathbf{R}^n$, $c, d \in \mathbf{R}^n$, $\alpha, \beta \in \mathbf{R}$, with $d^T x + \beta > 0$, $\forall x \in S$, or $d^T x + \beta < 0$, $\forall x \in S$, is both pseudoconvex and pseudoconcave function on S.

Definition 12.3.3 (Pseudolinear Function). *A function $f : S \subseteq \mathbf{R}^n \to \mathbf{R}$ is called a pseudolinear function on S if it is both a pseudoconvex function and a pseudoconcave function on S.*

For example, the ratio of the two linear functions described above as θ is a pseudolinear function on S.

We shift our attention to explore the relationship between the class of pseudoconvex functions and the class of quasiconvex functions.

We have already seen that the function $f(x) = x^3$, $x \in \mathbf{R}$, is a quasiconvex function on $\mathbf{R}$ but not a pseudoconvex function on $\mathbf{R}$. Thus, we can easily assert that a quasiconvex function need not be a pseudoconvex function. We shall now try to answer the converse.

Theorem 12.3.2 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be a pseudoconvex function on S. Then f is also a quasiconvex function on S.*

Proof. Let $x, u \in S$ be such that $f(x) \leq f(u)$. In order to prove that f is a quasiconvex function we need to prove that

$$f(\lambda x + (1-\lambda)u) \leq f(u), \quad \forall \lambda \in (0,1).$$

Contrary to this, suppose there exists $\bar{\lambda} \in (0,1)$ such that

$$f(\bar{\lambda} x + (1-\bar{\lambda})u) > f(u). \tag{12.4}$$

Set $\bar{x} = \bar{\lambda} x + (1-\bar{\lambda})u$. Then, $f(\bar{x}) > f(u)$, which on using pseudoconvexity of f yields

$$\nabla f(\bar{x})^T (u - \bar{x}) < 0.$$

Now, $x - \bar{x} = -(1-\bar{\lambda})(u - x)$ and $u - \bar{x} = \bar{\lambda}(u - x)$. Consequently, we obtain

$$-\frac{\bar{\lambda}}{(1-\bar{\lambda})} \nabla f(\bar{x})^T (x - \bar{x}) < 0.$$

Since $\bar{\lambda} \in (0,1)$, hence,

$$\nabla f(\bar{x})^T (x - \bar{x}) < 0.$$

Again, invoking pseudoconvexity of f to get, $f(x) \geq f(\bar{x})$. This along with (12.4) implies

$$f(x) > f(u),$$

which contradicts the assumed hypothesis. Thus,

$$f(x) \leq f(u) \Rightarrow f(\lambda x + (1-\lambda)u) \leq f(u), \ \forall \lambda \in [0,1],$$

giving f is a quasiconvex function on S. □

Corollary 12.3.1 *Let $S \subseteq \mathbf{R}^n$ be an open convex set and $f : S \to \mathbf{R}$ be a differentiable pseudoconcave function on S. Then f is also a quasiconcave function on S.*

Corollary 12.3.2 *A pseudolinear function $f : S \to \mathbf{R}$ on S is a quasilinear function on S.*

We now introduce two more classes of functions, namely, strictly pseudoconvex functions and strictly pseudoconcave functions.

Definition 12.3.4 (Strictly Pseudoconvex Function). *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$ be a differentiable function on S. Then f is called a strictly pseudoconvex function on S if*

$$x, u \in S, \ x \neq u, \ \nabla f(u)^T (x-u) \geq 0 \Rightarrow f(x) > f(u).$$

Equivalently,

$$x, u \in S, \ x \neq u, \ f(x) \leq f(u) \Rightarrow \nabla f(u)^T (x-u) < 0.$$

Definition 12.3.5 (Strictly Pseudoconcave Function). *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$ be a differentiable function on S. Then f is called a strictly pseudoconcave function on S if*

$$x, u \in S, \ x \neq u, \ \nabla f(u)^T (x-u) \leq 0 \Rightarrow f(x) < f(u).$$

Equivalently,

$$x, u \in S, \ x \neq u, \ f(x) \geq f(u) \Rightarrow \nabla f(u)^T (x-u) > 0.$$

It is important to take note of the following

(i) A strictly pseudoconvex (respectively, pseudoconcave) function on S is a pseudoconvex (respectively, pseudoconcave) function on S. But the reverse implication need not be true. Consider a function $f : \mathbf{R}^2 \to \mathbf{R}$ given by $f(x,y) = x^3 + x - y^2$. f is a pseudoconvex function on $\mathbf{R}^2$ as it does not possess a stationary point, but f is not a strictly pseudoconvex function on $\mathbf{R}^2$ as for $x = (1,1)$, $u = (1,-1)$, $x \neq u$, $f(x) = f(u)$ but $\nabla f(u)^T (x-u) > 0$. Also note that f is neither a convex function nor a concave function on $\mathbf{R}^2$ as the Hessian matrix $\nabla^2 f(x,y) = \begin{pmatrix} 6x & 0 \\ 0 & 2 \end{pmatrix}$ is neither a positive definite matrix nor a negative definite matrix on $\mathbf{R}^2$.

(ii) One can observe that if u is a stationary point of a strictly pseudoconvex (respectively, pseudoconcave) function, i.e. $\nabla f(u) = 0$, then u is the unique global minimizer (respectively, maximizer) of f on S. This means to say that the graph of a strictly pseudoconvex (respectively, pseudoconcave) function can not have more than one 'valley' (respectively, 'peak') with the same lowest depth (respectively, height).

In the theorem below, we shall see that every strictly pseudoconvex function is a strictly quasiconvex function.

Theorem 12.3.3 *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$ be a strictly pseudoconvex function on S. Then f is a strictly quasiconvex function on S.*

Proof. Suppose f is not a strictly quasiconvex function. Then there exist $x, u \in S$, $x \neq u$ with $f(x) \leq f(u)$ and $\bar{\lambda} \in (0, 1)$ such that

$$f(\bar{\lambda}x + (1 - \bar{\lambda})u) \geq f(u).$$

Writing $\bar{x} = \bar{\lambda}x + (1-\bar{\lambda})u$, we have, $\bar{x} \neq u$ and $f(u) \leq f(\bar{x})$. Using strict pseudoconvexity of f, it follows that

$$\nabla f(\bar{x})^T(u - \bar{x}) < 0.$$

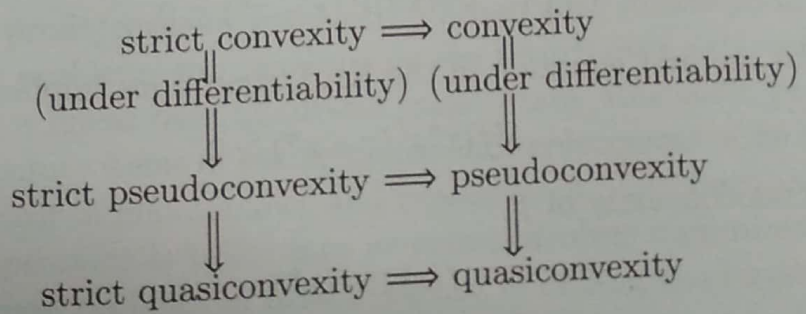
Now, $u - \bar{x} = -\dfrac{\bar{\lambda}}{1 - \bar{\lambda}}(x - \bar{x})$, hence,

$$\nabla f(\bar{x})^T(x - \bar{x}) > 0, \; x \neq \bar{x},$$

which, again by strict pseudoconvexity implies $f(x) > f(\bar{x})$, giving, $f(x) > f(u)$. This is contrary to the assumption that $f(x) \leq f(u)$. Therefore, f must be a strictly quasiconvex function on S. □

Corollary 12.3.3 *Let $S \subseteq \mathbf{R}^n$ be an open set and $f : S \to \mathbf{R}$ be a strictly pseudoconcave function on S. Then f is a strictly quasiconcave function on S.*

Relationship between convexity and its various generalizations is summarized as follows.



We now turn our attention to explore the role of quasiconvexity and pseudoconvexity in context to the optimality conditions for the nonlinear programming problems.

Recall the constrained nonlinear optimization problem

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \le 0 \quad (i = 1, \ldots, m), \end{aligned} \tag{12.5}$$

$f : \mathbf{R}^n \to \mathbf{R}$, $g_i : \mathbf{R}^n \to \mathbf{R}$ $(i = 1, \ldots, m)$ are differentiable functions on $\mathbf{R}^n$.

We have seen in Chapter 8 that the KKT optimality conditions, which otherwise are only the necessary optimality conditions for a point $x^\star \in \mathbf{R}^n$ to be an optimal solution of (12.5), also become 'sufficient optimality conditions' under the convexity hypothesis. Here, we shall be investigating whether the convexity assumption can be replaced by the weaker assumption of generalized convexity.

Theorem 12.3.4 *Let $x^\star$ be a feasible solution of problem (12.5), and there exists $\lambda^\star \in \mathbf{R}^m$ such that the following KKT optimality conditions hold*

$$\begin{aligned} \nabla f(x^\star) + \lambda^{\star T} \nabla g(x^\star) &= 0 \\ \lambda^{\star T} g(x^\star) &= 0 \\ \lambda^\star &\ge 0. \end{aligned}$$

Let $I(x^\star) = \{i : g_i(x^\star) = 0\}$ be the set of active constraints at $x^\star$. Suppose g_i, $i \in I(x^\star)$, are quasiconvex functions and f is a pseudoconvex function. Then $x^\star$ is an optimal solution of (12.5).

Proof. Let x be any other feasible solution of (12.5). Then,

$$g_i(x) \le 0 = g_i(x^\star), \quad i \in I(x^\star).$$

Using quasiconvexity hypothesis and $\lambda^\star \ge 0$, we get

$$\Big(\sum_{i \in I(x^\star)} \lambda_i^\star \nabla g_i(x^\star) \Big)^T (x - x^\star) \le 0.$$

Now, for $i \notin I(x^\star)$, $\lambda_i^\star = 0$. Thus, the above inequality can be rewritten as

$$(\lambda^{\star T} \nabla g(x^\star))^T (x - x^\star) \le 0$$

which along with the first equation of the KKT conditions implies

$$\nabla f(x^\star)^T (x - x^\star) \ge 0.$$

By virtue of pseudoconvexity of f,

$$f(x) \ge f(x^\star).$$

Since x is an arbitrary feasible point of (12.5), thus, $x^\star$ is an optimal solution of (12.5). □

12.4 Fractional Programming Problems

In this section, we shall be studying an important class of nonlinear programming problems, namely, fractional programming problems, described as follows

$$\text{Min} \quad \frac{f(x)}{g(x)}$$
$$\text{subject to} \quad x \in S, \tag{12.6}$$

where S is prescribed by finite number of constraint functions, i.e. $S = \{x \in \mathbf{R}^n : h_i(x) \leq 0 \ (i = 1, \ldots, m)\}$, and $f, g, h_i : \mathbf{R}^n \to \mathbf{R}$ $(i = 1, \ldots, m)$. In order to have the ratio function $\dfrac{f(x)}{g(x)}$ well defined, we assume that, $g(x) > 0$, $\forall\, x \in S$. For the case when $g(x) < 0$, $\forall\, x \in S$, we consider the objective ratio of the form $\dfrac{-f(x)}{-g(x)}$ instead.

It is important to note that the assumption on the sign of g function is natural. In fact, in practice g is generally a continuous function, hence, if g changes its sign in the feasible set S then it will take the value zero at least once in S. In other words, if there exist $x_1, x_2 \in S$ with $g(x_1) > 0$ and $g(x_2) < 0$, then by continuity of g, there exists $\hat{x} \in S$ such that $g(\hat{x}) = 0$. In such a scenario, the ratio $\dfrac{f(x)}{g(x)}$ is not appropriately defined on S.

If all the functions f, g, h_i $(i = 1, \ldots, m)$, involved in problem (12.6) are linear functions then the fractional programming problem is called the *linear fractional programming problem* (LFPP) else it is called the *nonlinear fractional programming problem.* Before proceeding with the theoretical developments and solution methods related to the fractional programs, we wish to outline some applications of fractional programming problems.

During the modeling of many industrial problems, like a stock cutting problem, it has been found useful to consider minimizing the ratio of the waste material and the used amount of raw material instead of simply minimizing the waste material alone. This type of formulation leads to linear fractional programming problems. On the other hand, in problems of resources allocation, it is obviously beneficial to consider maximization of the profit/cost or profit/revenue ratio rather than just maximizing the profit. In portfolio optimization problems too maximizing the return over risk ratio is important leading to optimizing a linear over quadratic ratio. Many stochastic processes, like, a periodic review of inventory items or maintenance and replacement of an item, where the ratio of the expected cost of maintenance and replacement of an item and the expected time between two inspections is taken into account, involves minimization of the cost to time ratio. A common problem in matrix theory is finding the largest eigenvalue of the given matrix which is basically maximizing the ratio of two quadratic functions. In communication models, the capacity of a communication channel is generally described

as the maximal transmission rate over all probabilities leading to nonlinear fractional programs. In nutshell we can say that fractional programming problems are frequently encountered in various fields thereby making their study important.

At this point one may critically raised the question as to why so much emphasis on studying fractional programming problems? Can not the class of fractional programming problems be treated just like other class of nonlinear optimization problems? These queries are very natural and need to be addressed before we proceed. One major reason for giving a special treatment to fractional programming problems is that even though, in general, these problems are non convex yet for certain specific types of nonlinear fractional programming problems the structure of the objective function allow us to design special algorithms for them.

In many cases, the numerator and the denominator of the objective function are found to be, respectively, a convex and a concave function. An extra nonnegativity condition, viz., $f(x) \geq 0,\ \forall x \in S$, when g is not a linear function on S, is generally imposed. This condition ensures that the convex-concave objective ratio $\dfrac{f(x)}{g(x)}$ is pseudoconvex and thus a quasiconvex function on S (recall, Theorem 12.3.1 and Theorem 12.3.2). In this case, a local min point of the fractional program (12.6) is also its global min point.

12.5 Linear Fractional Programming Problems

In this section, we shall be discussing two algorithms, namely, the *Charnes and Cooper algorithm* and the *simplex algorithm* to solve the following linear fractional programming problem

$$\begin{aligned} \text{Max} \quad & z = \frac{c^T x + \alpha}{d^T x + \beta} \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{12.7}$$

Here, $x, c, d \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, $\alpha, \beta \in \mathbf{R}$.

The feasible solution set $S = \{x \in \mathbf{R}^n : Ax = b,\ x \geq 0\}$ is assumed to be non empty and bounded. Furthermore, let $d^T x + \beta \neq 0,\ \forall x \in S$. As already discussed, for the validity of the ratio, we assume that the denominator function $d^T x + \beta$ keeps the same sign in S. If $\text{sign}(\underset{x \in S}{Min}\ (d^T x + \beta)) > 0$ then $d^T x + \beta > 0,\ \forall x \in S$, and if $\text{sign}(\underset{x \in S}{Max} d^T x + \beta) < 0$ then $d^T x + \beta < 0,\ \forall x \in S$. Without loss of generality, we take, $d^T x + \beta > 0,\ \forall x \in S$.

Charnes and Cooper Algorithm

In this algorithm, a variable transformation is used to convert problem (12.7) into a linear programming problem while preserving the geometry of the problem. For this, letting $d^T x + \beta = \frac{1}{w}$. The problem can be rewritten as

$$\begin{aligned} \text{Max} \quad & z = (c^T x + \alpha)\, w \\ \text{subject to} \quad & x \in S. \end{aligned}$$

Using a transformation $y = wx$, the above problem becomes

$$\begin{aligned} \text{Max} \quad & z = c^T y + \alpha w \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} Ay - bw &= 0 \\ d^T y + \beta w &= 1 \\ y, w &\geq 0. \end{aligned} \tag{12.8}$$

It may be noted here that the transformed problem (12.8) is a linear programming problem with an additional variable and one extra constraint as compared to problem (12.7).

The feasible solution space of problem (12.8) is denoted by $S^1 = \{(y, w) : Ay - bw = 0,\ d^T y + \beta w = 1,\ y \geq 0,\ w \geq 0\}$. We next prove some results regarding the two feasible sets S and S^1.

Result 12.5.1 *If $(y, w) \in S^1$ then $w > 0$.*

Proof. Suppose $(y, 0) \in S^1$. Then, $Ay = 0$ and $d^T y = 1$. Therefore, $y \neq 0$. Let $x \in S$, and $\mu > 0$ be an arbitrary real number. Then

$$A(x + \mu y) = Ax + \mu Ay = b,$$

$$x + \mu y \geq 0, \text{ with atleast one positive component.}$$

Thus, $x + \mu y \in S,\ \forall\, \mu > 0$. This contradicts the boundedness assumption on S. Consequently, $w > 0,\ \forall\, (y, w) \in S^1$. □

Result 12.5.2 *There is one to one correspondence between the feasible solutions of (12.7) and (12.8) with equal objective values.*

Proof. Let $x \in S$. Set $w = \dfrac{1}{d^T x + \beta} > 0$ and $y = wx$. Then, $(y, w) \in S^1$. Conversely, if $(y, w) \in S^1$ then by the previous result $w > 0$, and hence, $x = \dfrac{y}{w} \in S$.

Moreover, the objective values of problems (12.7) and (12.8) are equal at the two feasible solutions as

$$\frac{c^T x + \alpha}{d^T x + \beta} = c^T y + \alpha w.$$

□

Result 12.5.3 *There is one to one correspondence between the extreme points of the sets S and S^1.*

Proof. Let $x \in S$ be an extreme point of S. The corresponding feasible point of (12.8) is $y = wx$, $w = \dfrac{1}{d^T x + \beta}$.

If $(y, w) \in S^1$ is not an extreme point of S^1 then there exist two distinct points $(y_1, w_1) \in S^1$ and $(y_2, w_2) \in S^1$ such that for some λ, $0 < \lambda < 1$,

$$(y, w) = \lambda(y_1, w_1) + (1 - \lambda)(y_2, w_2)$$

$$\Rightarrow x = \frac{y}{w} = \frac{\lambda y_1 + (1 - \lambda) y_2}{w}$$

$$= \bar{\lambda} x_1 + (1 - \bar{\lambda}) x_2,$$

where $0 < \bar{\lambda} = \dfrac{\lambda w_1}{w} < 1$ and $x_1 = \dfrac{y_1}{w_1} \in S$, $x_2 = \dfrac{y_2}{w_2} \in S$, $x_1 \neq x_2$.

This implies that x is not an extreme point of S, leading to a contradiction. Hence, (y, w) must be an extreme point of S^1. The converse of the proof, i.e. if $(y, w) \in S^1$ is an extreme point of S^1 then the corresponding feasible point $x = \dfrac{y}{w} \in S$ is an extreme point of S, can be derived on the same lines. □

Result 12.5.4 *There is one to one correspondence between the optimal solutions of problems (12.7) and (12.8).*

Proof. Problem (12.8) is a linear program, so its optimal solution is one of the extreme point of S^1. Now, observe that because of Theorem 12.2.5, the objective function of (12.7) is a quasiconcave function on the polytope S. It then follows from Theorem 12.2.4 that the optimal solution of (12.7) is also obtained at one of the extreme point of S, say $\hat{x}$. Further, invoking Results 12.5.3 and 12.5.2, there exists a corresponding extreme point $(\hat{y}, \hat{w})$ of S^1 with objective value of (12.7) at $\hat{x}$ equals the objective value of (12.8) at $(\hat{y}, \hat{w})$. Moreover, since $\hat{x}$ is an optimal solution of (12.7), thus, by virtue of Results 12.5.2 and 12.5.3, $(\hat{y}, \hat{w})$ is an optimal solution of problem (12.8). □

The above result lies at the root of the Charnes and Cooper algorithm. From this, one can easily see that solving (12.7) is equivalent to solving a linear programming problem (12.8).

We now present an example to illustrate the working of this algorithm.

Example 12.5.1 *Solve the following linear fractional programming problem by the Charnes and Cooper algorithm*

$$\begin{aligned} \textit{Max} \quad & z = \frac{x_1}{x_1 + x_2 + 1} \\ \textit{subject to} \quad & \\ & x_1 + x_2 \leq 1 \\ & x_1 + 2x_2 \leq 1 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Solution Note that $x_1 + x_2 + 1 > 0$, for all (x_1, x_2) feasible for the problem. Applying the Charnes and Cooper transformation, $w = \dfrac{1}{x_1 + x_2 + 1}$ and $y_1 = wx_1$, $y_2 = wx_2$, the problem becomes

$$\begin{aligned} \text{Max} \quad & z = y_1 \\ \text{subject to} \quad & \\ & y_1 + y_2 - w \leq 0 \\ & y_1 + 2y_2 - w \leq 0 \\ & y_1 + y_2 + w = 1 \\ & y_1, y_2, w \geq 0. \end{aligned}$$

Introduce the slack variables $s_1, s_2 \geq 0$, the problem reduces to

$$\begin{aligned} \text{Max} \quad & z = y_1 \\ \text{subject to} \quad & \\ & y_1 + y_2 - w + s_1 = 0 \\ & y_1 + 2y_2 - w + s_2 = 0 \\ & y_1 + y_2 + w = 1 \\ & y_1, y_2, w, s_1, s_2 \geq 0. \end{aligned}$$

Solving the above problem by the two phase method, the optimal table is given by

	$y^{(1)}$	$y^{(2)}$	$y^{(w)}$	$y^{(s_1)}$	$y^{(s_2)}$
$y_1 = 1/2$	1	1	0	1/2	0
$s_2 = 0$	0	1	0	-1	1
$w = 1/2$	0	0	1	-1/2	0
$z_B = 1/2$	0	1	0	1/2	0

Therefore, the optimal value is $\frac{1}{2}$ with the optimal solution $y^\star = (\frac{1}{2}, 0)$ and $w^\star = \frac{1}{2}$. Thus, the optimal solution of the original linear fractional program is $x_1^\star = \dfrac{y_1^\star}{w^\star} = 1$, $x_2^\star = \dfrac{y_2^\star}{w^\star} = 0$ and the optimal value is $\dfrac{1}{2}$.

Simplex Algorithm for Linear Fractional Programming

As already observed that the objective function of (12.7) is a quasiconcave function and the feasible set S is a polytope, so, the optimal solution of (12.7) is obtained at an extreme point of S. Taking motivation from this simple fact, we study the simplex algorithm for problem (12.7) without restoring to any variable transformation. The simplex algorithm for linear programming problem has already been explained in detail in the first few chapters of this book. We assume that the readers are familiar and well versed with the notations and interpretations of all the concepts used in linear programming problems. We carry forward the same notations while describing the simplex algorithm for (12.7). Moreover, many steps in the present algorithm follows on the similar lines, so, without going into the detailed reasoning we present only their outline.

We start with an assumption that a basic feasible solution (b.f.s.) with basis B is available. Therefore, $x_B = B^{-1}b$. Set

$$V_N = c_B^T x_B + \alpha, \quad V_D = d_B^T x_B + \beta.$$

The value of the objective function at x_B is $z = \dfrac{V_N}{V_D}$.

Let $a^{(j)}$ be a column in a matrix A that is not in B. Then, since B is a basis, $a^{(j)}$ can be expressed as a linear combination of the columns of B matrix, i.e.

$$a^{(j)} = By_j = \sum_{i=1}^{m} y_{ij} b^{(i)}.$$

Set $z_j^1 = c_B^T y_j$, $z_j^2 = d_B^T y_j$.

We first find the condition that ensures that the current b.f.s. can be improved to get another b.f.s. with improved objective value.

Suppose $\bar{B}$ is a new basis obtained from B by replacing a column $b^{(r)}$ of B by $a^{(j)}$. Therefore, the columns of $\bar{B}$ are given by $\bar{b}^{(r)} = a^{(j)}$, $\bar{b}^{(i)} = b^{(i)}$ $(i \neq r)$. The leaving variable x_{B_r} is chosen according to the minimum ratio test

$$\theta_j = Min\left\{\frac{x_{B_i}}{y_{ij}} \;:\; y_{ij} > 0\right\} = \frac{x_{B_r}}{y_{rj}} \qquad \text{(say)}.$$

The new basic variables $\bar{x}_{B_r}$ are given by

$$\bar{x}_{B_i} = x_{B_i} - \theta_j y_{ij} \quad (i = 1, \ldots, m,\ i \neq r),$$
$$\bar{x}_{B_r} = \theta_j.$$

Let the new value of the objective function be $\bar{z} = \dfrac{\overline{V}_N^1}{\overline{V}_D^2}$, where

$$\overline{V}_N^1 = V_N - \theta_j(z_j^1 - c_j) \quad \text{and} \quad \overline{V}_D^2 = V_D - \theta_j(z_j^2 - d_j).$$

We pause here to again emphasize that the detailed working of all the above expressions is explained in Chapter 3.

The objective function value will strictly improve if $\bar{z} > z$, i.e.

$$\frac{\overline{V}_N^1}{\overline{V}_D^2} - \frac{V_N}{V_D} > 0,$$

which on simplification yields

$$\theta_j\{V_N(z_j^2 - d_j) - V_D(z_j^1 - c_j)\} > 0.$$

Letting $\Delta_j = V_N(z_j^2 - d_j) - V_D(z_j^1 - c_j)$, we obtain, $\theta_j\Delta_j > 0$.

For $\theta_j > 0$, the improvement in the objective value z is possible if $\Delta_j > 0$. Thus, the current b.f.s. is optimal for (12.7) if $\Delta_j \leq 0$, for all those variables x_j which are currently nonbasic variables. Note that for basic variables, we already have, $\Delta_j = 0$. Thus, the optimality criteria (for maximization problem (12.7)) is that $\Delta_j \leq 0$ $(j = 1, \ldots, n)$.

Suppose $\theta_j = 0$. It mean $x_{B_r} = 0$ and $\bar{z} = z$. This implies that there is no improvement in the objective function value. Here x_{B_r}, which is a basic variable with value zero, becomes a nonbasic variable and the new variable $\bar{x}_{B_r}$ becomes a basic variable with value zero, the values of all the other variables remain the same, leading to the situation of degenerate b.f.s. In this case, the basis changes but the corresponding extreme points remains the same. The case of degeneracy in b.f.s. has already been discussed in Chapter 3.

The above discussion is summarized in the following theorem.

Theorem 12.5.1 *If all $\Delta_j \leq 0$ then the current b.f.s. x_B is an optimal solution of the linear fractional programming problem (12.7).*

Remark 12.5.1 *If some $\Delta_j > 0$ and for that $y_{ij} \leq 0$, $\forall i$, then the linear fractional programming problem (12.7) has unbounded solution, which contradicts the boundedness of the feasible set S.*

Remark 12.5.2 *The objective function of (12.7), $\dfrac{c^T x + \alpha}{d^T x + \beta}$, is a pseudolinear function, hence, a local optimizer is a global optimizer. Consequently, the b.f.s. x_B with all $\Delta_j \leq 0$ is the global optimal solution of problem (12.7).*

The mechanism of the simplex algorithm for linear fractional programming problem is now illustrated through an example.

Example 12.5.2 *Solve the following linear fractional programming problem by the simplex algorithm*

$$\text{Max} \quad z = \frac{x_1}{x_1 + x_2 + 1}$$

subject to

$$x_1 + x_2 \leq 1$$
$$x_1 + 2x_2 \leq 1$$
$$x_1, x_2 \geq 0.$$

Solution Introducing the slack variables $s_1, s_2 \geq 0$, the problem reduces to

$$\text{Max} \quad z = \frac{x_1}{x_1 + x_2 + 1}$$

subject to

$$x_1 + x_2 + s_1 = 1$$
$$x_1 + 2x_2 + s_2 = 1$$
$$x_1, x_2, s_1, s_2 \geq 0.$$

Solving the problem by the simplex method we get following tableaus

	$y^{(1)}$	$y^{(2)}$	$y^{(s_1)}$	$y^{(s_2)}$
$\leftarrow s_1 = 1$	$\boxed{1}$	1	1	0
$s_2 = 1$	1	2	0	1
$z_j^1 - c_j$	-1	0	0	0
$z_j^2 - d_j$	-1	-1	0	0
$z_B = 0,\ \Delta_j \rightarrow$	1	0	0	0
	$\uparrow$			

	$y^{(1)}$	$y^{(2)}$	$y^{(s_1)}$	$y^{(s_2)}$
$x_1 = 1$	1	1	1	0
$s_2 = 0$	0	1	-1	1
$z_j^1 - c_j$	0	1	1	0
$z_j^2 - d_j$	0	0	1	0
$z_B = 1/2,\ \Delta_j \rightarrow$	0	-2	-1	0

Since $\Delta_j \leq 0,\ \forall\, j$, thus the optimality criteria is satisfied. The optimal solution of the given problem is $x_1^\star = 1,\ x_2^\star = 0$ and the optimal value is $\frac{1}{2}$.

12.6 Nonlinear Fractional Programming Problems

In the previous section, we have concentrated on studying the solution methodologies for linear fractional programming problems. In this section we move ahead to study

a solution procedure for a subclass of the class of *nonlinear fractional programming problems*.

The class of nonlinear fractional programming problems is too large to be solved by a single algorithm. Thus, when we say a subclass we actually mean a particular class of nonlinear fractional programming problems in which the numerators are concave functions and the denominators are positive convex functions. Such nonlinear fractional programming problems are called concave-convex fractional programming problems. Note that for such problems the objective ratio is a pseudoconcave function.

The algorithm presented below is called the *Dinkelbach's algorithm*, named after Werner Dinkelbach of Germany who first described this algorithm in 1967. Recall that a general nonlinear fractional programming problem is given by

$$\begin{aligned} &\text{Max} && \frac{f(x)}{g(x)} \\ &\text{subject to} \end{aligned}$$

$$x \in S = \{x \in \mathbf{R}^n : h_i(x) \leq 0 \ (i = 1, \ldots, m)\}. \tag{12.9}$$

We assume that S is a non empty compact convex set and $f, g : \mathbf{R}^n \to \mathbf{R}$ are continuous functions on S with $g(x) > 0$, $\forall\, x \in S$.

For instance the assumption that S is a convex set is true if each h_i $(i = 1, \ldots, m)$, is a quasiconvex function on $\mathbf{R}^n$, as $S = \bigcap_{i=1}^{m} \{0 - \text{level set of } h_i\}$. Also, S is a closed set if each h_i $(i = 1, \ldots, m)$ is a continuous function on $\mathbf{R}^n$.

The algorithm for solving (12.9) shall make use of the following auxiliary problem with parameter $q \in \mathbf{R}$.

$$\begin{aligned} &\text{Max} && f(x) - qg(x) \\ &\text{subject to} && x \in S. \end{aligned} \tag{12.10}$$

Observe that if f is concave, g is convex and $q \geq 0$, then $f - qg$ is a concave function, whereas, $\dfrac{f}{g}$ is not a concave function. This makes (12.10) easier to solve than (12.9). Moreover, f and g are continuous functions on a compact convex set S, hence the two problems (12.9) and (12.10) possess optimal solutions in S.

Denote by $F(q)$ the optimum objective value of (12.10), i.e.

$$F(q) = Max\{f(x) - qg(x) : x \in S\},\ q \in \mathbf{R}.$$

The function F has some nice properties which we would like to share with the readers.

Result 12.6.1 *F is a convex function on* $\mathbf{R}$.

Proof. Let $q_1, q_2 \in \mathbf{R}$, $\lambda \in [0, 1]$. Taking $q = \lambda q_1 + (1 - \lambda) q_2$. Let $x_q \in S$ be the max point of $F(q)$, i.e.

$$\begin{aligned} F(q) &= Max\{f(x) - qg(x) \,:\, x \in S\} \\ &= f(x_q) - qg(x_q) \\ &= f(x_q) - (\lambda q_1 + (1-\lambda)q_2)g(x_q) \\ &= \lambda(f(x_q) - q_1 g(x_q)) + (1-\lambda)(f(x_q) - q_2 g(x_q)) \\ &\leq \lambda \underset{x\in S}{Max}(f(x) - q_1 g(x)) + (1-\lambda)\underset{x\in S}{Max}(f(x) - q_2 g(x)) \\ &= \lambda F(q_1) + (1-\lambda)F(q_2). \end{aligned}$$

Thus,

$$F(\lambda q_1 + (1-\lambda)q_2) \leq \lambda F(q_1) + (1-\lambda)F(x_2),\ \forall\, \lambda \in [0,1],\ \forall\, q_1, q_2 \in \mathbf{R}.$$

This completes the proof. □

Result 12.6.2 *F is a continuous function on* $\mathbf{R}$.

Proof. Using Result 12.6.1 along with the fact that a convex function defined on an open convex set (here, it is $\mathbf{R}$) is continuous in its domain, we get the desired result. □

Result 12.6.3 *F is a strictly monotonic decreasing function on* $\mathbf{R}$.

Proof. Let $q_1, q_2 \in \mathbf{R}$ with $q_1 < q_2$. Suppose $x_2 \in S$ is the point where $F(q_2)$ attains its maximum value. Then,

$$\begin{aligned} F(q_2) &= f(x_2) - q_2 g(x_2) \\ &< f(x_2) - q_1 g(x_2) \\ &\leq F(q_1), \end{aligned}$$

where the strict inequality follows on account of $q_1 < q_2$ and $g(x_2) > 0$. Thus,

$$q_1 < q_2 \Rightarrow F(q_2) < F(q_1).$$

Thereby, implying that F is a monotonic decreasing function on $\mathbf{R}$. □

Result 12.6.4 *The nonlinear equation $F(q) = 0$ has the unique solution in* $\mathbf{R}$.

Proof follows by virtue of Result 12.6.2 and Result 12.6.3. □

Result 12.6.5 *Let $x^\star \in S$ and $q^\star = \dfrac{f(x^\star)}{g(x^\star)}$. Then $F(q^\star) \geq 0$.*

Proof. We have

$$\begin{aligned} F(q^\star) &= Max\{f(x) - q^\star g(x) \,:\, x \in S\} \\ &\geq f(x^\star) - q^\star g(x^\star) = 0. \end{aligned}$$

□

Theorem 12.6.1 *$x^\star \in S$ is an optimal solution of (12.9) if and only if $x^\star$ is an optimal solution of $F(q^\star)$ with optimal objective value $F(q^\star) = 0$, where, $q^\star = \dfrac{f(x^\star)}{g(x^\star)}$.*

Proof. Suppose $x^\star$ is an optimal solution of problem (12.9). Then,

$$q^\star = \frac{f(x^\star)}{g(x^\star)} \geq \frac{f(x)}{g(x)}, \quad \forall x \in S$$

implying

$$f(x) - q^\star g(x) \leq f(x^\star) - q^\star g(x^\star), \quad \forall x \in S.$$

Consequently, $x^\star$ is an optimal solution of the problem

$$F(q^\star) = Max\{f(x) - q^\star g(x) : x \in S\},$$

with $F(q^\star) = 0$.

The converse follows by tracing the steps backward.

Similar theorem can be stated and proved for the minimization case as well. □

Theorem 12.6.2 *$x^\star \in S$ is an optimal solution of $Min\left\{\frac{f(x)}{g(x)} : x \in S\right\}$ if and only if $x^\star$ is an optimal solution of $Min\{f(x) - q^\star g(x) : x \in S\}$ with optimal objective value zero, where, $q^\star = \frac{f(x^\star)}{g(x^\star)}$.*

Dinkelbach's Algorithm

As a consequence of the above results and theorems, it follows that there is a correspondence between the optimal solutions of the nonlinear fractional programming problem (12.9) and the nonlinear parametric programming problem (12.10). Taking clue from this, a mechanism is developed which solves the nonlinear parametric programming problem (12.10) that in turn provides an optimal solution of the original nonlinear fractional programming problem (12.9).

In view of Theorem 12.6.1, solving (12.9) is equivalent to finding the root of the equation $F(q) = 0$ which, on account of Result 12.6.4, is unique. An iterative scheme is proposed to achieve this aim.

We begin the algorithm with $q_0 = 0$ (or we can start with any other value of q with $F(q) \geq 0$).

$$F(q_0) = Max\{f(x) : x \in S\} \geq 0.$$

This problem is constrained convex program, hence, its optimal solution can be found by applying appropriate nonlinear convex optimization technique. Suppose x_0 is an optimal solution of this problem.

Two cases arise, either $F(q_0) = 0$ or $F(q_0) > 0$.

If $F(q_0) = 0$ then x_0 is an optimal solution of (12.9) and the process terminates.

Suppose $F(q_0) > 0$. Set $k \leftarrow 0$. Let

$$q_{k+1} = \frac{f(x_k)}{g(x_k)} > q_k,$$

and solve

$$F(q_{k+1}) = Max\{f(x) - q_k g(x) : x \in S\} \geq 0.$$

Let x_{k+1} be an optimal solution of this convex problem (which can be determined by some numerical technique). Then

$$F(q_{k+1}) = f(x_{k+1}) - q_k g(x_{k+1}) \geq f(x_k) - q_k g(x_k) = 0.$$

Two cases can be discussed.

If $F(q_{k+1}) = 0$ then x_{k+1} is an optimal solution of (12.9), and so stop the procedure.

Else if, $F(q_{k+1}) > 0$ then set $k \leftarrow k+1$ and continue.

It is important to take note of the following points for implementation convenience of the Dinkelbach's algorithm.

(i) From computational view point, it is convenient and acceptable to pre decide the tolerance $\delta > 0$, and then if $F(q_k) < \delta$, (instead $F(q_k) = 0$ exactly), the procedure can be stopped. If $F(q_k) \geq \delta$, then the procedure is continued.

(ii) At each iteration, a constrained convex optimization problem

$$F(q_k) = Max\{f(x) - q_k g(x) : x \in S\} \tag{12.11}$$

is solved. Thus, it is sufficient to generate the KKT point of the problem. For the unconstrained case, when $S = \mathbf{R}^n$, it is equivalent to finding the stationary point of the problem provided the functions involved in the problem are differentiable. Note that an optimal solution of (12.11) is not necessarily unique. Furthermore, the sequence $\{q_k\}$ of parameters generated in the described procedure is not a unique sequence.

(iii) The sequence $\{q_k\}$ is strictly monotonic increasing sequence, i.e. $q_k < q_{k+1}$, $\forall k$, and the corresponding sequence $\{F(q_k)\}$ is strictly monotonic decreasing sequence, $0 < F(q_{k+1}) < F(q_k)$, $\forall k$, bounded from below by zero.

In the proposed algorithm, beginning with $q = 0$, the value of q is steadily increasing while the value of $F(q)$ is steadily decreasing. Since F is a continuous strictly monotonically decreasing function with the unique root of the equation $F(q) = 0$, as depicted graphically in Fig 12.4, the optimal solution $x^\star$ (up to the desired accuracy $\delta > 0$) with the corresponding point $q^\star$ will be achieved. It means that the algorithm converges to an (almost) optimal solution of (12.9).

Theorem 12.6.3 (Convergence of the algorithm). *Let $x^\star$ be an optimal solution of (12.9) and $q^\star = \dfrac{f(x^\star)}{g(x^\star)}$. Then the sequence $\{q_k\}$ generated by the proposed algorithm converges to $q^\star$.*

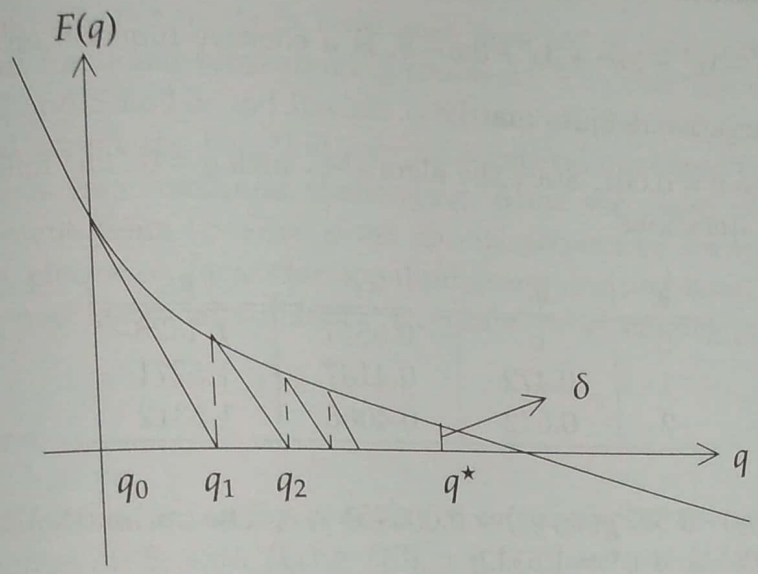


Fig. 12.4.

Proof. We first observe that $q^\star$ is an upper bound of the sequence $\{q_k\}$, i.e. $q_k \leq q^\star$, $\forall\, k$. This is because for $q > q^\star$, $F(q) < 0$, and the algorithm would have stopped much before this situation to arise.

Thus, $\{q_k\}$ is a monotonically increasing sequence bounded from above, hence, it must be convergent. Let

$$\lim_{k\to\infty} q_k = \hat{q}.$$

As F is a continuous function,

$$\lim_{k\to\infty} F(q_k) = F(\hat{q}).$$

By construction of the sequence in the algorithm, we get,

$$F(\hat{q}) = 0 = F(q^\star).$$

Invoking Result 12.5.4, $\hat{q} = q^\star$. Thus completing the proof. □

Example 12.6.1 *Solve the following nonlinear fractional program by the Dinkelbach's algorithm*

$$\text{Max} \quad \frac{-3x^2 - 2y^2 + 4x + 8y - 8}{x^2 + y^2 - 6y + 8}$$

subject to

$$x + 3y \leq 5$$
$$x,\ y \geq 0.$$

Solution Here, $g(x, y) = x^2 + y^2 - 6y + 8 = x^2 + (y-3)^2 - 1$, is a strictly convex function with $g(x, y) > 0$, $\forall\, (x, y) \in S$.

Also, $f(x,y) = -3x^2 - 2y^2 + 4x + 8y - 8$, is a concave function on S as $\nabla^2 f(x,y) = \begin{pmatrix} -6 & 0 \\ 0 & -4 \end{pmatrix}$ is a negative definite matrix.

Set the tolerance $\delta = 0.001$. Start the algorithm with $q = 0$. The following table gives the results for two iterations.

k	q_k	x_k	y_k
0	0	0.5517	1.4828
1	0.472	0.4187	1.5271
2	0.522	0.4066	1.5312

$F(0.522) = f(x_2, y_2) - 0.522\, g(x_2, y_2) = 0.000451 < \delta$. The optimal solution of the given problem is $x^\star = 0.4066$ and $y^\star = 1.5312$.

12.7 Summary and Additional Notes

- In Section 12.2 and Section 12.3, we introduced several new classes of generalized convex functions. Many important properties, including the local-global solution characterizations, are derived for the newly defined classes. Although we have carried out the in depth discussion on these classes of functions but for more reading refer to the excellent texts by Avriel et al. [6], Bazaraa and Shetty [11], Mangasarian [109], Martos [114]. The research articles by Ferland [56], Ponstein [129] and Greenberg and Pierskalla [70] are worth seeing.
- The last few decades have witnessed tremendous growth in the area of generalized convexity. Many more new classes of generalized convex functions have been introduced. Among them, perhaps the most widely studied class is that of invex functions introduced by Hanson [75] in 1981. The nonsmooth (i.e. nondifferentiable) variants of these generalizations are also very popular and well investigated. One can easily find enough materials on them. However, going by the flavor and the aim of this text, we have focussed on some very basic natural generalizations of convex functions.
- Section 12.4 highlights the importance and applications of fractional programming problems in various fields. To know more about this class of optimization problems one can refer to the text by Craven [40] and the survey articles by Stancu-Minasian [147] and Schaible and Ibaraki [141] appeared in research journals.
- In Section 12.5, the Charnes and Cooper variable transformation method and the simplex method are explained to solve linear fractional programming problems. These algorithms were initiated by Charnes and Cooper [35] and Swarup [153], respectively.
- The best source of reading for the last section, Section 12.6, is the original research article by Dinkelbach [48] himself. One can also refer to the book by Kambo [89].

- Several new and efficient methods have been reported in literature for solving some special types of fractional programming problems. The excellent survey by Stancu-Minasian [147] and Schaible and Ibaraki [141] provide good references in this regard.
- We realize and accept the fact that solving a general nonlinear fractional programming problem is very hard and challenging. Some successful attempts have been reported in recent years to solve some special structured fractional programming problems, like, piecewise-linear fractional programming problems, minmax fractional programming problems and stochastic fractional programming problems.

12.8 Exercises

12.1 *Let $S \subseteq \mathbf{R}^n$ be a nonempty convex set containing origin and $f : S \to \mathbf{R}$ be a quasiconvex function on S, with $f(x) \geq f(0)$, $x \in S$. Show that $f(\lambda x) \leq f(x)$, $\forall\, \lambda \in [0,1]$.*

12.2 *Let $S \subseteq \mathbf{R}^n$ be a nonempty convex set. Show that the function $f : S \to \mathbf{R}$ is quasiconvex on S if and only if for every $x_1, x_2 \in S$, the one-dimensional function $g : [0,1] \to \mathbf{R}$ defined as, $g(\lambda) = f(\lambda x_1 + (1-\lambda)x_2)$, is quasiconvex on $[0,1]$.*

12.3 *Let $f(x) = \frac{1}{2}x^T Q x + c^T x$, Q is an $n \times n$ symmetric matrix. Show that f is convex on $\mathbf{R}^n$ if and only if it is quasiconvex on $\mathbf{R}^n$. Is the above statement true if we restrict the domain of f to $\mathbf{R}^n_+$, i.e. $x \in \mathbf{R}^n_+$? Justify your arguments.*

12.4 *Let $\{f_\gamma : \gamma \in I\}$ be an arbitrary family of quasiconvex functions on $\mathbf{R}^n$, and let $g(x) = \sup_{\gamma \in I} f_\gamma(x)$. Show that g is a quasiconvex function on $\mathbf{R}^n$.*

12.5 *For $\gamma > 0$, consider the following family of functions,*

$$f_\gamma(x) = \begin{cases} x^3 + \gamma(x-10), & -1 \leq x \leq 0 \\ \gamma(x-10), & 0 \leq x \leq 1 \\ (x-1)^3 + \gamma(x-10), & 1 \leq x \leq 2. \end{cases}$$

For a fixed $\gamma > 0$, is f_γ a quasiconvex or a pseudoconvex function on $[-1,2]$? Let $g(x) = \sup_{\gamma > 0} f_\gamma(x)$. Is g a quasiconvex or a strictly quasiconvex function on $[-1,2]$? Is g a pseudoconvex function on $[-1,2]$? Justify your answers.

12.6 *Let f be a positive quasiconcave function defined over a convex subset S of $\mathbf{R}^n$. Show that $g(x) = 1/f(x)$ is a quasiconvex function on S. What can you say about the reciprocal function, $1/f$, if f is a positive quasiconvex function?*

12.7 *Let $h : \mathbf{R}^n \to \mathbf{R}$ be a quasiconvex function and let $g : \mathbf{R} \to \mathbf{R}$ be a nondecreasing function. Show that the composite function, $f = g \circ h : \mathbf{R}^n \to \mathbf{R}$, is a quasiconvex function.*

12.8 *Let $S \subseteq \mathbf{R}^n$ be a nonempty convex set and $g, h : S \to \mathbf{R}$. Define a function $f : S \to \mathbf{R}$ by $f(x) = g(x)h(x)$. Show that f is quasiconvex if the following two conditions hold*

(i) g is convex on S and $g(x) \leq 0$, $\forall x \in S$
(ii) h is concave on S and $h(x) > 0$, $\forall x \in S$.

12.9 *Let $S \subseteq \mathbf{R}^n$ be a nonempty open convex set. Consider a function $f : S \to \mathbf{R}$ with $f(x) > 0$, $\forall x \in S$. Show that if $\ln(f)$ is a concave function on S then f is a pseudoconcave function on S.*

12.10 *Solve the following linear fractional programming problems by both the Charnes and Cooper method and the simplex method*

(i) *Max* $z = \dfrac{2x_1 + x_2}{-x_1 - 2x_2}$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 6 \\ 2x_1 + x_2 &\geq 4 \\ x_1, x_2 &\geq 0. \end{aligned}$$

(ii) *Max* $z = \dfrac{2x_1 + 3x_2}{x_1 + x_2 + 7}$

subject to

$$\begin{aligned} 3x_1 + 5x_2 &\leq 15 \\ 4x_1 + 3x_2 &\leq 12 \\ x_1, x_2 &\geq 0. \end{aligned}$$

(iii) *Max* $z = \dfrac{x_2 - 5}{-x_1 - x_2 + 9}$

subject to

$$\begin{aligned} 2x_1 + 5x_2 &\geq 10 \\ 4x_1 + 3x_2 &\leq 20 \\ -x_1 + x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

12.11 *Using the variable transformation, associate a linear programming problem with the linear fractional programming problem (LFPP)*

Min $\dfrac{c^T x + c_o}{d^T x + d_o}$

subject to

$$\begin{aligned} Ax &\geq 0 \\ x &\geq 0. \end{aligned}$$

Let $(y^\star, w^\star)$ be an optimal solution of the linear programming problem. If $w^\star \neq 0$ then show that $x^\star = y^\star/w^\star$ is an optimal solution of the (LFPP) and the objective functions values of the (LFPP) and the corresponding linear program are equal.

12.12 *Suppose $w^\star = 0$ in the Charnes and Cooper transformation. Prove that the feasible set of (LFPP) is unbounded.*

12.13 *Solve the following fractional programming problems graphically*

$$(i) \qquad \underset{(x_1, x_2) \in S}{Min} \quad \frac{6x_1 + 2x_2 + 6}{3x_1 + 5x_2 + 3}$$

$$(ii) \qquad \underset{(x_1, x_2) \in S}{Min} \quad \frac{6x_1 + 2x_2 + 6}{3x_1 + 5x_2 + 3}$$

where $S = \{(x_1, x_2) : x_1 - x_2 \leq 2,\ 2x_1 - x_2 \leq 6,\ x_1, x_2 \geq 0\}$. Which of (i) and (ii) has an alternate optimal solution? Justify your answer.

12.14 *Solve the following fractional programming problems by the Dinkelbach's method*

(i) *Max* $z = \dfrac{5x_1 + 3x_2}{5x_1 + 2x_2 + 1}$

subject to

$$\begin{aligned} 3x_1 + 5x_2 &\leq 15 \\ 5x_1 + 2x_2 &\geq 10 \\ x_1, x_2 &\geq 0. \end{aligned}$$

(ii) *Max* $z = \dfrac{2x_1 + 2x_2 + 1}{x_1^2 + x_2^2 + 3}$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 3 \\ x_1, x_2 &\geq 0. \end{aligned}$$

(iii) *Max* $z = \dfrac{x_1 + 2x_2}{3x_1^2 + x_2^2 + 1}$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 - x_2 &\leq 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

13
Multi-objective Optimization: Theory and Methods

13.1 Introduction

By now we are well versed with the concepts of linear and nonlinear programming problems involving optimization of a single objective function. We now take a step further and enter in the area where the programming problems require optimization of more than one objective function. To get comfortable with this idea, we begin our discussion by considering a very simple decision making problem of buying a car. Let us pose a question to ourself. What features we will look for in a car of our choice? Cost, comfort, space, mileage, safety, engine power, height from the ground level, power steering, color, and may be some additional technical features, like, power windows, quality of AC, and so on. From this list it is obvious that a decision of buying a particular car is not based on a single criterion, of say cost alone, but many more criteria are equally vital in the final decision. Problems of this kind can not be modeled through a single objective optimization problems. One needs to look beyond that and talk about *multiobjective programming* (MOP). As the name is self explanatory, a multiobjective programming problem (MOPP) deals with optimization of more than one objective criterion. MOPPs are frequently encountered in problems of product design, management decisions, resource planning, to name a few. This branch of optimization is also known by other names, like, vector optimization, multicriteria optimization, multiattribute optimization and so forth.

In this chapter, we shall be concentrating on MOPPs unlike the previous chapters where the focus was on a single objective optimization problems. We shall be describing and characterizing various solution concepts associated with the class of MOPPs. In the later part of the chapter, we shall be discussing a technique, known as *goal programming technique*, to solve a particular class of linear multiobjective programming problems. Our aim is to get familiar with the otherwise vast and complex topic of MOP. We therefore shall be avoiding the technical complexities and restrict ourselves to understand the fundamental ideas involved in the context.

13.2 Conflicting Objectives and Tradeoff

Going back to the problem of buying a car, it is evident that some of the several listed criteria are conflicting in nature, like, more comfort can generally be achieved by increasing the cost of a car, more advance technical features will also enhance the cost of a car, luxury car provides more comfort but generally gives less mileage. In practice too, MOPP involves several conflicting and non-commensurate objective functions that have to be optimized simultaneously over a feasible region.

For instance, consider two objectives represented by functions, say, $f_1(x) = x^2$ and $f_2(x) = (x-1)^2$. We wish to minimize them simultaneously over the set $[0,1]$.

We may sketch the graphs of $f_1(x)$ and $f_2(x)$, and observe that while f_1 is increasing in $[0,1]$, f_2 is decreasing in $[0,1]$, thereby resulting in a situation where it is not possible to find an $x \in [0,1]$ that can minimize both the objectives.

If we can find a feasible vector $x^\star$ that optimizes all the objective criteria simultaneously then we have certainly achieved an ideal solution of the problem. But quite often, improvement in one criterion results in a loss in another criterion leading to the unlikely existence of an ideal solution. It can be seen as some kind of tradeoff between various objectives. Visualizing and resolving the tradeoffs is one of the key aspect of MOP. For this reason, one has to look for the 'best' compromise solution. Now, 'best' can be defined differently in different situations. In economics, 'best' is referred to the decisions taken by the buyers and sellers or the governments which simultaneously optimize several criteria. One of the most frequently quoted example thereof is Taxation. An optimal collected tax is one which maximizes the revenue for common goods while maintaining sufficient incentives for individuals to earn reasonably good income from their work.

Not surprisingly, the first person to describe such a tradeoff was an Economist F. Y. Edgeworth . He defines an optimum within the context of two consumers criteria, P and Q, as "It is required to find a point (x, y) such that in whatever direction we take an infinitely small step, P and Q do not increase together but that, while one increases, the other decreases". Vilfredo Pareto, chair of political economy at the University of Lausanne, Switzerland, wrote in 1906, "The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimations".

The above statements beautifully capture the balance of the socio-economic structure of any society. The very same principle is applicable to MOPPs. One needs to create a balance between various conflicting objective functions to arrive at the 'best' or 'optimal' solution in some appropriate sense. Of course, we need to know how to designate a particular feasible point to be a 'best' compromise solution. We shall be examining this issue from mathematical viewpoint in the sections to follow.

13.3 Various Solution Concepts

The following form of MOPP is studied in this chapter

$$\underset{x \in S}{Min}\ f(x) = (f_1(x), \ldots, f_p(x)). \tag{13.1}$$

Here we assume that S is a nonempty subset of $\mathbf{R}^n$ representing the feasible set of problem (13.1), and $f : S \to \mathbf{R}^p$ is a given vector function comprising of p objective criteria to be minimized.

For $p = 1$, problem (13.1) reduces to a scalar nonlinear programming problem which has been a subject of study in the previous chapters. So, for the multiobjective case, we take $p \geq 2$. Moreover, it is not necessary that all the p objective criteria are to be minimized, some criteria may involve maximization process. For instance, in a car buying problem discussed earlier, we would like to maximize comfort, maximize mileage and minimize cost of a car. Actually in context of modeling the problem it does not matter whether we investigate minimization or maximization problem. One can convert all the maximization criteria into the minimization form by using the identity, $Max f_i(x) = -Min(-f_i(x))$. With this understanding, we study MOPP in the minimization form in problem (13.1). Observe that we have yet to define what we mean by minimization of a vector objective function, $f(x) \in \mathbf{R}^p$ with $x \in S$, in (13.1).

The basic problem to realize here is that unlike the real space $\mathbf{R}$, the space $\mathbf{R}^p$, $p \geq 2$, is not an ordered space unless we define an appropriate partial order. It simply means that given any two distinct real numbers x and y it is always possible to determine the greater among them. But the same is not true if x and y are vectors in $\mathbf{R}^p$. For instance, it is not possible to compare the vectors, $(2,1)^T$ and $(1,2)^T$, in general.

Now, for $x \in S$, we get the resultant objective vector $f(x)$ in $\mathbf{R}^p$. Let $f(S) = \{y \in \mathbf{R}^p : \exists x \in S \text{ such that } y = f(x)\}$ denotes the image set of S under f. To define an optimal solution in the sense of minimization we need to compare vectors in the image set $f(S)$, and for this, we need to identify a partial order relation in $\mathbf{R}^p$.

To appreciate this aspect of MOPP, we consider the problem

$$\underset{x \in [0,1]}{Min}\ (x^2, (x-1)^2)$$

Let $y_1 = x^2$ and $y_2 = (x-1)^2$. For $y = (y_1, y_2) \in f(S)$, there exists $x \in S$ such that $f(x) = y$, i.e. $y_1 = x^2$, $y_2 = (x-1)^2$. As $x \in S = [0,1]$, $y_1 \geq 0$, $y_2 \geq 0$. Moreover, $x = \sqrt{y_1}$ and $(x-1) = \sqrt{y_2}$ yields the set, $f(S) = \{(y_1, y_2) : y_1 \geq 0, y_2 \geq 0, \sqrt{y_1} + \sqrt{y_2} = 1\}$.

Now if we choose any two vectors in the set $f(S)$, say $(1,0)$ and $(0,1)$, or $(\frac{1}{4}, \frac{1}{4})$ and $(\frac{9}{16}, \frac{1}{16})$, we can not decide as which one is greater of the two unless some priorities are attached with the two objective functions.

Thus the major thrust in defining the solution concepts for MOPP (13.1) lies in construction of an appropriate partial order in the space $\mathbf{R}^p$.

In this chapter we assume that $\mathbf{R}^p$ is partially ordered by a binary relation induced by $\mathbf{R}^p_+$, the nonnegative orthant of $\mathbf{R}^p$. By this we mean the following. For $x, y \in \mathbf{R}^p$,

$$\begin{aligned} x \leqq_{\mathbf{R}^p_+} y &\Leftrightarrow y - x \in \mathbf{R}^p_+; \\ x \leq_{\mathbf{R}^p_+} y &\Leftrightarrow y - x \in \mathbf{R}^p_+ \setminus \{0\}; \\ x <_{\mathbf{R}^p_+} y &\Leftrightarrow y - x \in int\mathbf{R}^p_+. \end{aligned}$$

Thus, $\begin{pmatrix} 1 \\ 2 \end{pmatrix} <_{\mathbf{R}^2_+} \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ while $\begin{pmatrix} 1 \\ 2 \end{pmatrix} \leq_{\mathbf{R}^2_+} \begin{pmatrix} 1 \\ 3 \end{pmatrix}$.

It is important to note the difference between $x \leqq_{\mathbf{R}^p_+} y$ and $x \leq_{\mathbf{R}^p_+} y$. While the first one means $x_i \leqq y_i$ $(i = 1, \ldots, p)$, the latter one imply $x_i \leqq y_i$ $(i = 1, \ldots, p, i \neq j)$ and $x_j < y_j$, for some j. From now onwards we suppress the subscript $\mathbf{R}^p_+$ in the above relations. Moreover, for $x, y \in \mathbf{R}$, we continue to use $x \leq y$ to denote x is less than equal to y. The partial order $\leq$ is to be understood in the right sense in the given context.

We are now ready to introduce two solution concepts for MOPP (13.1).

Definition 13.3.1 (Weak Efficient Solution). *$x^\star \in S$ is called a weak efficient solution of problem (13.1) if there does not exist $x \in S$ such that $f(x) < f(x^\star)$. In other words, it says that whenever $x \in S$, $f(x) - f(x^\star) \notin -(int\mathbf{R}^p_+)$. In set notation form the same can be explained as $(f(S) - f(x^\star)) \bigcap (-int\mathbf{R}^p_+) = \emptyset$.*

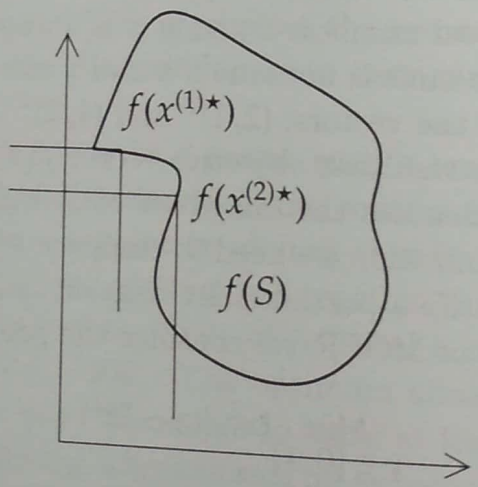


Fig. 13.1.

Remark 13.3.1 *In two or three dimensional spaces, we can graphical illustrate the concept of weak efficiency. Let $p = 2$, i.e, $f(S) \subseteq \mathbf{R}^2$. The above definition suggests that on shifting the origin to a point $f(x^\star)$, if we find that $int\mathbf{R}^2_- (= -int\mathbf{R}^2_+)$ does not intersect the set $f(S)$, then $x^\star$ is a weak efficient solution of (13.1), see Fig 13.1. $x^{(1)\star}$ is a weak efficient solution of some minimization problem represented by $f(S)$ whereas $x^{(2)\star}$ is not a weak efficient solution of the same problem.*

We present below few examples in support of above definition. The examples also illustrate that the weak efficient solution of MOPP is not necessarily unique.

Example 13.3.1 *Let* $S = \{x = (x_1, x_2) : x_1+x_2 \geq 2, (x_1-1)^2+(x_2-1)^2 \leq 4, x_1, x_2$ *are integers* $\}$ *and* $f : S \to \mathbf{R}^2$ *be defined as* $f(x_1, x_2) = (x_1+x_2^2,\ x_1^2+x_2)$. *Consider the problem* $\underset{x \in S}{Min}\ f(x)$, *and identify the set of weak efficient solutions.*

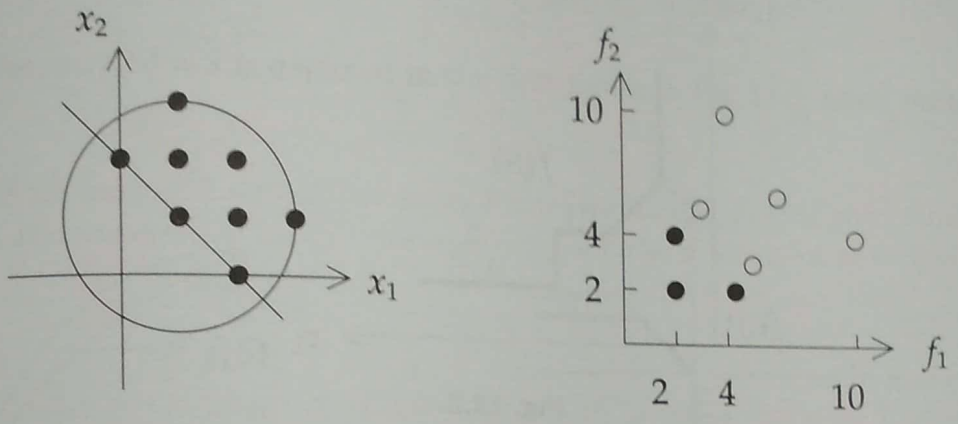


Fig. 13.2.

Solution Observe that $S = \{(0,2), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (3,1)\}$, consequently, $f(S) = \{(4,2), (2,2), (5,3), (10,4), (2,4), (3,5), (6,6), (4,10)\}$.

The sets S, $f(S)$ are shown in Fig 13.2. The set of weak efficient solutions of the problem is given by $\{(0,2), (1,1), (2,0)\}$.

Example 13.3.2 *Let* $S = [0,1] \times [0,1]$ *and* $f : S \to \mathbf{R}^2$ *be the identity map given by* $f(x_1, x_2) = (x_1, x_2)$. *Consider the problem*

$$\underset{x \in S}{Min}\ f(x), \tag{13.2}$$

and obtain the set of weak efficient solutions.

Solution Obviously $f(S) = S$. $x^\star = (0,1) \in S$ is a weak efficient solution of (13.2), as there is no $x \in S$ with $x_1 < 0$ and $x_2 < 1$, i.e. $f(x) < f(x^\star)$. Similarly for $x^\star = (1,0) \in S$, there is no $x \in S$ with $x_1 < 1$ and $x_2 < 0$, in other words, there exists no $x \in S$ with $f(x) < f(x^\star)$. Consequently, $(1,0)$ is also a weak efficient solution of (13.2). In fact it can easily be seen that all the points in the set $\{(x_1, 0) : 0 \leq x_1 \leq 1\} \bigcup \{(0, x_2) : 0 \leq x_2 \leq 1\}$ are weak efficient solutions of (13.2).

Example 13.3.3 *Let* $S = \mathbf{R}^2$ *and* $f : S \to \mathbf{R}^2$ *be defined as* $f(x_1, x_2) = (x_1^2, x_2^2)$. *Obtain the set of weak efficient solutions of* $\underset{x \in S}{Min}\ f(x)$.

Solution It is clear that $f(S) = \mathbf{R}^2_+$. Thus the set of weak efficient solutions of the problem is given by $(\{0\} \times \mathbf{R}) \bigcup (\mathbf{R} \times \{0\})$.

Example 13.3.4 *Let* $S = \{x = (x_1, x_2) : x_1 \geq 1, x_2 \geq 1, x_1 + x_2 \geq 4, \max\{2x_1, 3x_2\} \geq 6\}$ *and* $f : S \to \mathbf{R}^2$ *be* $f(x_1, x_2) = (x_1, x_2)$. *Obtain the set of weak efficient solution of* $\underset{x \in S}{Min}\ f(x)$.

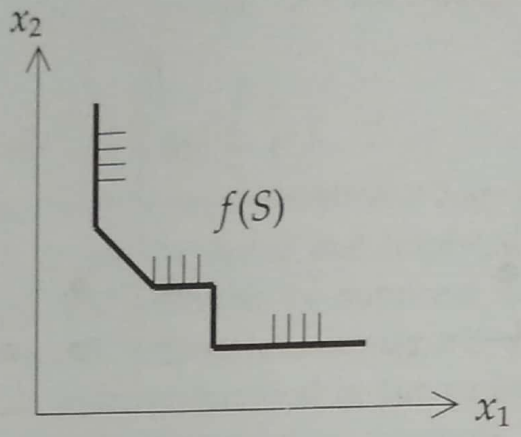


Fig. 13.3.

Solution The set $f(S)$ is depicted in Fig 13.3. The set of weak efficient solutions is marked by a bold curve.

We now define another solution concept called efficient solution of MOPP (13.1). The idea of efficiency is based upon the conviction that no criterion can be improved without worsening at least one other criterion.

Definition 13.3.2 (Efficient Solution). *$x^\star \in S$ is called an efficient solution of problem (13.1) if there does not exist $x \in S$ such that $f(x) \leq f(x^\star)$. In other words, it says that whenever $x \in S$, $f(x) - f(x^\star) \notin -(\mathbf{R}^p_+ \setminus \{0\})$. In set notation form the same can be explained as $(f(S) - f(x^\star)) \bigcap (-\mathbf{R}^p_+ \setminus \{0\}) = \emptyset$.*

This solution is also known by the names *Pareto solution* or *non-inferior solution.*

Definition 13.3.3 (Efficient Frontier). *The set of efficient solutions of MOPP (13.1) is called efficient frontier.*

Remark 13.3.2 *It follows from Definition 13.3.1 and Definition 13.3.2 that every efficient solution is a weak efficient solution of problem (13.1). But the converse need not hold. Some of the following examples justify this fact.*

Example 13.3.5 *Let* $S = \{x = (0, x_2) : x_2 \leq 0\}$ *and* $f : S \to \mathbf{R}^2$ *be* $f(x_1, x_2) = (x_1, x_2)$. *Consider* $\underset{x \in S}{Min}\ f(x)$, *and identify the set of efficient solutions.*

Solution Here $f(S) = S$. The set of efficient solutions is an empty set because for any $(x_1^\star, x_2^\star) \in S$, we can always find another $(x_1, x_2) \in S$ with $x_1 = x_1^\star = 0$ and $x_2 < x_2^\star$. Note that the set of weak efficient solutions is the entire set S.

Going back to Example 13.3.1, we find that the problem has only one efficient solution, namely, $(1,1)$. Also, recall Example 13.3.2. It is clear that $(0,0)$ is the only efficient solution of the MOPP considered therein while the set of weak efficient solutions is an uncountable subset of S.

Example 13.3.6 *Let* $S = \{x = (x_1, x_2) : x_1 \in [0,1],\ \sqrt{x_1} \le x_2 \le \sqrt[3]{x_1}\}$. *Identify the sets of weak efficient and efficient solutions of* $\underset{x \in S}{Min}\ f(x_1, x_2) = (x_1, x_2)$.

Solution The set $f(S) = S$ is depicted in the first figure in Fig 13.4. Here, both the set

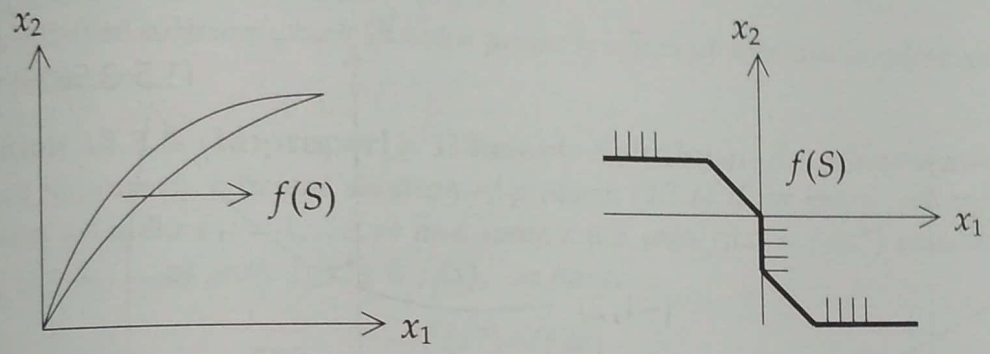


Fig. 13.4.

of weak efficient solutions and the set of efficient solutions are equal to the singleton set $\{(0,0)\}$.

However, we need to realize that a MOPP can have many efficient solutions. For instance, in Example 13.3.4, the efficient frontier is $\{(1-\lambda)(1,3) + \lambda(2,2) : \lambda \in [0,1]\} \bigcup \{(3,1)\}$. The following examples also confirm the statement.

Example 13.3.7 *Let* $S = \{(x_1, x_2) : x_1 \ge 1\} \bigcup \{(x_1, x_2) : x_1 \le 0,\ x_2 \ge 1\} \bigcup \{(x_1, x_2) : x_1 \le 0,\ x_2 \le 1,\ x_1 + x_2 \ge 0\} \bigcup \{(x_1, x_2) : 0 \le x_1 \le 1,\ x_1 + x_2 \ge -1\}$. *Consider* $\underset{(x_1, x_2) \in S}{Min}\ f(x_1, x_2) = (x_1, x_2)$, *and sketch its efficient frontier.*

Solution The graphical illustration of the set $f(S) = S$ is presented in the second figure in Fig 13.4. The efficient frontier is described by

$$\{(1-\lambda)(-1,1) : \lambda \in (0,1)\} \cup \{(1-\lambda)(0,-1) + \lambda(1,-2) : \lambda \in [0,1]\}.$$

Note that the efficient frontier is not a connected set. Moreover, if instead we have considered the MOPP, $Min_{(x_1,x_2) \in S} f(x_1, x_2) = (-x_2, x_1)$, then the set of efficient solutions is an empty set.

In an example to follow, the efficient frontier of MOPP is connected.

Example 13.3.8 *Consider the multiobjective programming problem*

$$\begin{aligned} &\textit{Min} && f(x_1, x_2) = (-x_1,\ x_1 + x_2^2) \\ &\textit{subject to} \\ & && x_1^2 - x_2 \leq 0 \\ & && x_1 + 2x_2 \leq 3, \end{aligned}$$

and sketch its efficient frontier.

Solution The feasible set S and its image set $f(S)$ are shown in Fig 13.5.

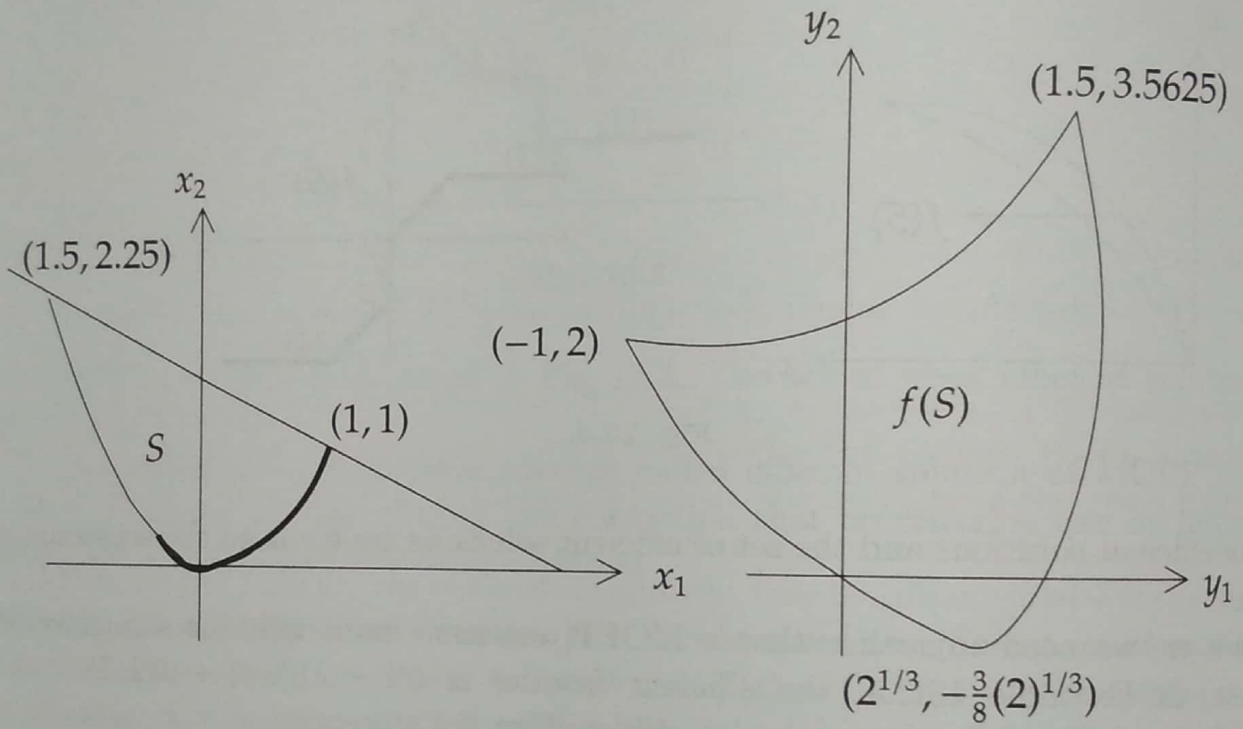


Fig. 13.5.

The efficient frontier is given by $\{(x_1, x_2) : x_1 \in \left[-\frac{1}{2}\sqrt[3]{2},\ 1\right],\ x_2 = x_1^2\}$.

Through the above examples it is clear that the efficient solution of MOPP (13.1) is one for which if one criterion improves than at least one other criterion deteriorate in the sense of minimization. In other words, if value of one criterion decreases at some feasible point than the value of at least one other criterion increases at the same feasible point. However, it is possible that improvement in one criterion is marginal as compared to the deterioration in the other criterion leading to anomalous efficient solutions. To exclude such efficient solutions, Arthur M. Geoffrion [67] introduced a sharper notion of solution called properly efficient solution. The idea behind the proper efficiency is to eliminate unbounded tradeoffs between various criteria.

Definition 13.3.4 (Properly Efficient Solution). *$x^\star \in S$ is called a properly efficient solution of problem (13.1) if $x^\star$ is an efficient solution of problem (13.1), and there*

exists a real number $M > 0$ such that for every index i $(i = 1, \ldots, p)$ and every $x \in S$ with $f_i(x) < f_i(x^\star)$, there exists at least one index j $(j = 1, \ldots, p)$, such that $f_j(x^\star) < f_j(x)$ and

$$\frac{f_i(x^\star) - f_i(x)}{f_j(x) - f_j(x^\star)} \leq M.$$

A properly efficient solution is sometimes refer to as *Geoffrion properly efficient solution* or *properly Edgeworth-Pareto optimal solution*. We shall restrict ourself to call it a properly efficient solution.

Remark 13.3.3 *(i) It follows from the definition that every properly efficient solution is an efficient solution of problem (13.1).*
(ii) An efficient solution which is not a properly efficient solution is called an improperly efficient solution.

Definition 13.3.5 (Improperly Efficient Solution). *An efficient solution $x^\star$ is called an improperly efficient solution of problem (13.1) if for every real number $M > 0$, there exist an index i $(i = 1, \ldots, p)$ and some $x \in S$ with $f_i(x) < f_i(x^\star)$ such that for every index j $(j = 1, \ldots, p)$ with $f_j(x^\star) < f_j(x)$, we have,*

$$\frac{f_i(x^\star) - f_i(x)}{f_j(x) - f_j(x^\star)} > M.$$

The commonsense reasoning says that improperly efficient solutions are not desired as improvement in one criterion comes only with a large sacrifice in the other criterion.

Example 13.3.9 *Consider the linear multiobjective programming problem*

$$\begin{aligned} &\textit{Min} && f(x_1, x_2) = (x_1 + 2x_2,\ 2x_1 - x_2) \\ &\textit{subject to} \\ &&& x_1 + x_2 \leq 1 \\ &&& x_1, x_2 \geq 0. \end{aligned}$$

Identify the set of efficient solutions and show that $x^\star = (0, 1)$ is a properly efficient solution of the given MOPP.

Solution Let $y_1 = x_1 + 2x_2, y_2 = 2x_1 - x_2$. Then the feasible set S gets transformed to the set $f(S) = \{(y_1, y_2) : 3y_1 + y_2 \leq 5, y_1 + 2y_2 \geq 0, 2y_1 - y_2 \geq 0\}$, shown in Fig 13.6.

It is clear that the set of efficient solutions is given by $\{(0, x_2) : x_2 \in [0, 1]\}$. Consider an efficient solution $x^\star = (0, 1)$. Then $f(x^\star) = (2, -1)$, and for any $x \in S$, $x \neq x^\star$, $y_1 < 2$, $y_2 > -1$. Take $i = 1$ and $j = 2$ in the definition of properly efficient solution. It can easily be verified that

$$\frac{f_1(x^\star) - f_1(x)}{f_2(x) - f_2(x^\star)} = \frac{2 - y_1}{y_2 + 1} \leq 2.$$

Thus, $(0, 1)$ is a properly efficient solution of the MOPP.

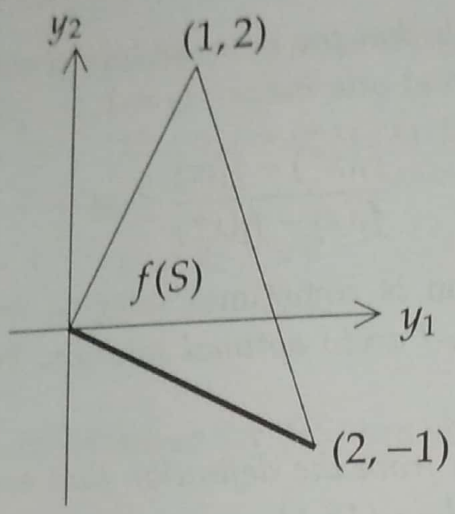


Fig. 13.6.

Example 13.3.10 *Let* $S = \{x = (x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$ *and* $f : S \to \mathbf{R}^2$ *be the identity map,* $f(x_1, x_2) = (x_1, x_2)$. *Consider the MOPP:* $\underset{x \in S}{\text{Min}}\, f(x_1, x_2)$. *Identify the set of efficient solutions. Are* $(0, -1)$ *and* $(-1, 0)$ *properly efficient solutions? Justify your answer.*

Solution The set of efficient solutions is $\left\{(x_1, x_2) : x_1 \in [-1, 0],\ x_2 = -\sqrt{1 - x_1^2}\right\}$. Consider an efficient solution $x^\star = (0, -1)$. For any natural number n, the point $x_n = (-\frac{1}{n}\sqrt{2n-1}, -1 + \frac{1}{n}) \in S$, and we have,

$$f_1(x_n) = -\frac{1}{n}\sqrt{2n-1} < f_1(x^\star), \quad f_2(x_n) = -1 + \frac{1}{n} > f_2(x^\star).$$

Consequently,

$$\frac{f_1(x^\star) - f_1(x)}{f_2(x) - f_2(x^\star)} = \sqrt{2n-1} \to \infty \text{ as } n \to \infty.$$

Thus $(0, -1)$ is an improperly efficient solution of MOPP. In fact we can show that $(-1, 0)$ is also an improperly efficient solution of MOPP while all other efficient solutions are properly efficient solutions of MOPP.

Remark 13.3.4 *There are several variants of properly efficient solutions in literature with more general partial ordering and in more general vector spaces. Geoffrion's properly efficient solution is based on a natural partial ordering in* $\mathbf{R}^p$. *Its simplicity makes it easy to visualize and understand. For this reason we have included it in the text.*

The next section governs the characterization of properly efficient solution which in turn suggests a computational procedure for finding the properly efficient solutions of MOPP (13.1).

13.4 Weighted Sum Approach

For ready reference recall the MOPP

$$\underset{x \in S}{Min}\ f(x) = (f_1(x), \ldots, f_p(x)), \tag{13.3}$$

S is the feasible set and $f : S \to \mathbf{R}^p$, $p \geq 2$, is the objective function.

For arbitrary $\lambda_i > 0$ $(i = 1, \ldots, p)$, $\sum_{i=1}^{p} \lambda_i = 1$, associate a scalar problem with MOPP (13.3) as

$$\underset{x \in S}{Min} \sum_{i=1}^{p} \lambda_i f_i(x) \tag{13.4}$$

The scalar λ_i $(i = 1, \ldots, p)$ can be interpreted as some positive weight or priority assigned to the i-th objective criterion by the decision maker. The sum of all the weights is usually taken to be one to ensure that the criteria are appropriately scaled and relatively placed. We present a hypothetical situation for clarity. Suppose $p = 3$ and $\lambda_1 = \frac{1}{2}$, $\lambda_2 = \frac{1}{3}$, $\lambda_3 = \frac{1}{6}$. It means that the first objective criterion is 3 times important as compared to the third objective criterion and 1.5 times important in comparison with the second objective criterion, while the second objective criterion is twice important in comparison to the third objective criterion. Equivalently, the objectives priorities are set as 3:2:1 by the decision maker.

From now onwards we take,

$$\Lambda = \{\lambda = (\lambda_1, \ldots, \lambda_p)^T \in \mathbf{R}^p : \lambda_i > 0\ (i = 1, \ldots, p),\ \sum_{i=1}^{p} \lambda_i = 1\}.$$

The following two theorems relate the properly efficient solutions of MOPP (13.3) with the optimal solutions of the scalar nonlinear programming problem (13.4).

Theorem 13.4.1 *Let $x^\star \in S$ be an optimal solution of problem (13.4), for fixed $\lambda \in \Lambda$. Then $x^\star$ is a properly efficient solution of problem (13.3).*

Proof. The proof is achieved in two parts. First we prove that $x^\star$ is an efficient solution of problem (13.3), and later, proper efficiency of $x^\star$ for problem (13.3) is worked out. Both parts are proved by contradiction.

Suppose $x^\star$ is not an efficient solution of problem (13.3). Then there exists $x \in S$ and an index i, such that

$$f_j(x) \leq f_j(x^\star)\ \ (j = 1, \ldots, p,\ j \neq i), \quad \text{and} \quad f_i(x) < f_i(x^\star). \tag{13.5}$$

Since $\lambda_i > 0$ $(i = 1, \ldots, p)$, it follows from (13.5) that

$$\sum_{i=1}^{p} \lambda_i f_i(x) < \sum_{i=1}^{p} \lambda_i f_i(x^\star),$$

a contradiction to the optimality of $x^\star$ for problem (13.4).

Next, assume that $x^\star$ is an improperly efficient solution of problem (13.3). Choose $M = (p-1) \max_{i,j=1,\ldots,p} \frac{\lambda_j}{\lambda_i}$. Then, for some index i $(i = 1,\ldots,p)$ and some $x \in S$ with $f_i(x) < f_i(x^\star)$, we obtain,

$$\frac{f_i(x^\star) - f_i(x)}{f_j(x) - f_j(x^\star)} > M \quad (j = 1,\ldots,p) \text{ with } f_j(x) > f_j(x^\star).$$

This implies

$$f_i(x^\star) - f_i(x) > M(f_j(x) - f_j(x^\star)) \geq (p-1)\frac{\lambda_j}{\lambda_i}(f_j(x) - f_j(x^\star)) \quad (j = 1,\ldots,p,\ j \neq i).$$

Multiplying throughout by $\frac{\lambda_i}{p-1}$ and summing over all j $(j = 1,\ldots,p,\ j \neq i)$, yields

$$\lambda_i(f_i(x^\star) - f_i(x)) > \sum_{j=1,\, j\neq i}^{p} \lambda_j(f_j(x) - f_j(x^\star)).$$

Consequently,

$$\sum_{j=1}^{p} \lambda_j f_j(x) < \sum_{j=1}^{p} \lambda_j f_j(x^\star),$$

again contradicting optimality of $x^\star$ for problem (13.4). The conclusion is now evident from the two contradictions. □

Theorem 13.4.2 *Let S be a convex set and each f_i $(i = 1,\ldots,p)$ be a convex function on S. If $x^\star \in S$ is a properly efficient solution of problem (13.3) then there exists $\lambda \in \Lambda$ such that $x^\star$ is an optimal solution of problem (13.4) with this λ.*

We skip the proof here as it requires the knowledge of the 'separation theorem of convex sets'. However, we recommend the readers to the research article of Geoffrion [67] for the detailed proof.

Example 13.4.1 *Consider the two criteria linear MOPP*

$$\begin{aligned} \textit{Min} \quad & f(x_1, x_2) = (-2x_1 + x_2,\ x_1 - 2x_2) \\ \textit{subject to} \quad & \\ & -x_1 + x_2 \leq 2 \\ & 2x_1 - x_2 \leq 10 \\ & x_1 + x_2 \leq 8 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{13.6}$$

Construct the weighted sum LPP and analyze the same in the light of Theorem 13.4.1 and Theorem 13.4.2.

Solution Let $y_1 = -2x_1 + x_2$, $y_2 = x_1 - 2x_2$. The image set $f(S)$ is given by $\{(y_1, y_2) : y_1 - y_2 \le 6,\ y_1 \ge -10,\ y_1 + y_2 \ge -8,\ 2y_1 + y_2 \le 0,\ y_1 + 2y_2 \le 0\}$. Fig 13.7 shows the feasible set S and the set $f(S)$.

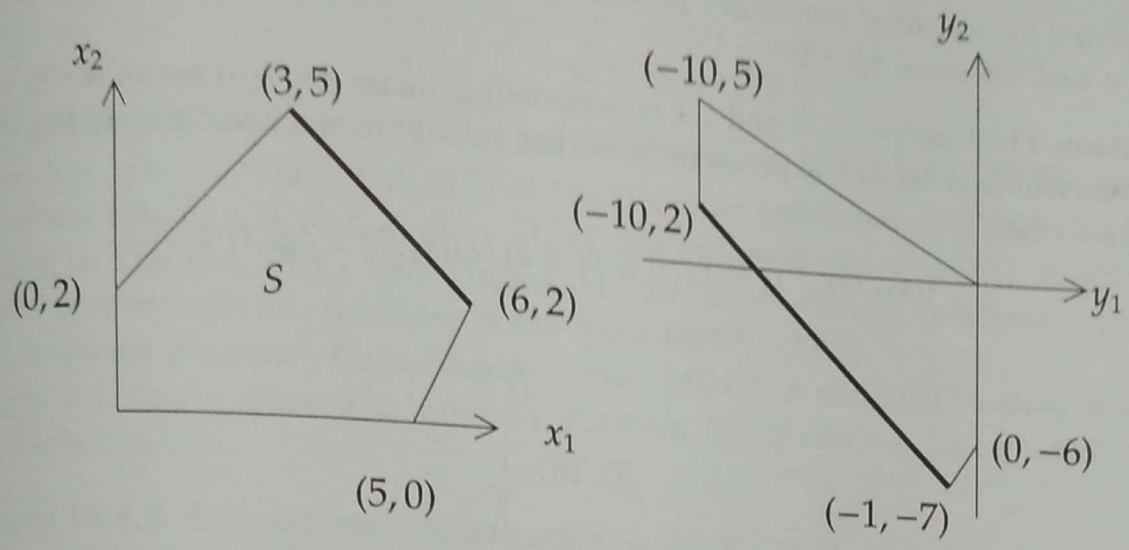


Fig. 13.7.

The efficient frontier is marked by bold line in the first figure in Fig 13.7, and is described by $\{\mu(3,5) + (1-\mu)(6,2) : \mu \in [0,1]\}$.

We formulate the weighted sum LPP with weights $(\lambda, 1-\lambda)$, $\lambda \in (0,1)$

$$\begin{aligned} \text{Min} \quad & (1-3\lambda)x_1 + (3\lambda - 2)x_2 \\ \text{subject to} \quad & \\ & -x_1 + x_2 \le 2 \\ & 2x_1 - x_2 \le 10 \\ & x_1 + x_2 \le 8 \\ & x_1, x_2 \ge 0. \end{aligned} \tag{13.7}$$

For $\lambda = \frac{1}{3}$, the optimal solution of (13.7) is $(3,5)$, while for $\lambda = \frac{2}{3}$, the optimal solution is $(6,2)$. For $\lambda = \frac{1}{2}$, the optimal solutions of the weighted linear program (13.7) generate the entire line segment connecting the points $(3,5)$ and $(6,2)$. Thus, on account of Theorem 13.4.1 and Theorem 13.4.2, all efficient solutions are properly efficient solutions of problem (13.7).

We next present an example of nonlinear convex MOPP to illustrate the weighted sum approach.

Example 13.4.2 *Consider the two criteria nonlinear MOPP*

$$\begin{aligned} \textit{Min} \quad & f(x_1, x_2) = (x_1^2 + x_2^2,\ (x_1 - 3)^2 + (x_2 - 3)^2) \\ \textit{subject to} \quad & (x_1 - 3)^2 + x_2^2 \leq 9 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 2. \end{aligned} \tag{13.8}$$

Construct the weighted sum scalar problem, and analyze the same in the light of Theorem 13.4.1 and Theorem 13.4.2.

Solution The feasible set S and the corresponding image set $f(S)$ are shown in Fig 13.8. The efficient frontier of the problem is marked in bold in the first figure in Fig 13.8, and it is described by

$$\{(x_1, x_2) : x_1 = x_2,\ 0 \leq x_1 \leq 2\} \cup \{(x_1, 2) : 2 \leq x_1 \leq 3\}.$$

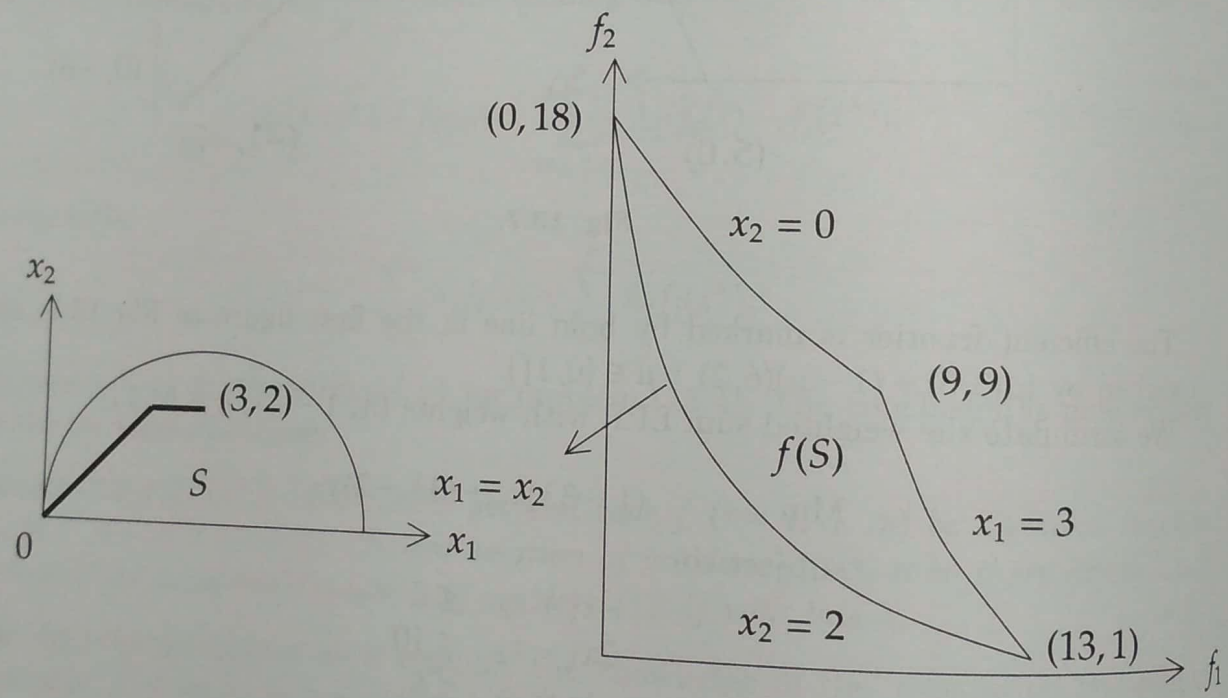


Fig. 13.8.

For weight vector $(\lambda, 1 - \lambda)$, $\lambda \in (0, 1)$, the weighted sum nonlinear convex programming problem is

$$\begin{aligned} \text{Min} \quad & f(x_1, x_2) = \lambda(x_1^2 + x_2^2) + (1 - \lambda)((x_1 - 3)^2 + (x_2 - 3)^2) \\ \text{subject to} \quad & (x_1 - 3)^2 + x_2^2 \leq 9 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 2. \end{aligned} \tag{13.9}$$

Writing the KKT conditions for problem (13.9), we get the following system of equations

$$\begin{aligned}
2x_1 + 6\lambda + 2\mu x_1 - 6\mu + \nu - s &= 6\\
2x_2 + 6\lambda + 2\mu x_2 + \eta - t &= 6\\
\mu((x_1-3)^2 + x_2^2 - 9) &= 0\\
\nu(x_1 - 3) &= 0\\
\eta(x_2 - 2) &= 0\\
sx_1 &= 0\\
tx_2 &= 0\\
\mu, \nu, \eta, s, t &\geq 0.
\end{aligned}$$

After simple calculations it can be shown that when $\lambda \in [\frac{1}{3}, 1)$, all the efficient solutions $\{(x_1, x_2) : x_1 = x_2,\ 0 < x_1 \leq 2\}$ of (13.8) can be generated, while for $\lambda \in (0, \frac{1}{3})$, the resultant set, $\{(x_1, 2) : 2 < x_1 < 3\}$, of efficient solutions of (13.8) is obtained. According to Theorem 13.4.1 and Theorem 13.4.2, these efficient solutions constitute the set of properly efficient solutions of the given MOPP (13.8).

The following example of a nonlinear convex MOPP is included to show that the condition of properly efficient solution in Theorem 13.4.2 can not be replaced by the efficient solution.

Example 13.4.3 *Consider the convex nonlinear MOPP in* **R**

$$\begin{aligned}
&Min \quad f(x) = (-x^2, x^3)\\
&subject\ to \quad x \geq 0.
\end{aligned} \tag{13.10}$$

Show that the Theorem 13.4.2 fails to hold at $x^\star = 0$.

Solution Clearly $x^\star = 0$ is an efficient solution of (13.10). But $x^\star$ is an improperly efficient solution because for any $M > 0$ there exists a feasible $x > 0$ such that $f_1(x) < f_1(x^\star)$ and $f_2(x) > f_2(x^\star)$ but, $\dfrac{f_1(x^\star) - f_1(x)}{f_2(x) - f_2(x^\star)} > M$.

Now suppose we construct a weighted scalar problem of (13.10) as

$$\begin{aligned}
&\text{Min} \quad \lambda_1(-x^2) + \lambda_2 x^3\\
&\text{subject to} \quad x \geq 0,
\end{aligned} \tag{13.11}$$

with $\lambda_1 > 0,\ \lambda_2 > 0$. Then it can be shown that $x^\star = 0$ is not an optimal solution of (13.11), for any $\lambda_1, \lambda_2 > 0$. For if, $x^\star$ is its optimal solution then we must have, $\lambda_1(-x^2) + \lambda_2 x^3 \geq 0,\ \forall\, x \geq 0$ yielding $-\lambda_1 + \lambda_2 x \geq 0,\ \forall\, x > 0$. The latter is possible only if $\lambda_1 = 0$.

Remark 13.4.1 *(i) The approach described above to study MOPP (13.3) is known as a weighted sum approach. This is because the multiobjective programming problem (13.3)*

has been converted into a scalar nonlinear programming problem (13.4) with the help of weighted mean of the objective functions f_i. $\lambda \in \Lambda$ in problem (13.4) is appropriately refer to as a weight vector.

(ii) In view of the Theorem 13.4.2, the weighted sum approach seems to be suitable for convex MOPP (13.3). However, for a general nonconvex MOPP, it is possible that not every properly efficient solution can be determined using this approach. Despite this, the approach works well for a large class of generalized convex MOPP with convex efficient frontier.

(iii) The above two theorems characterize properly efficient solutions of MOPP (13.3). Similar characterizations can be derived for weak efficient solutions and efficient solutions of MOPP (13.3). The main principle behind these characterizations is same. It relates various solutions of MOPP with optimal solutions of suitable scalar nonlinear programming problem. The difference emerges in the choice of the weight vector $\lambda \in \mathbf{R}^p$. We summarize the results in the following two theorems.

Theorem 13.4.3 *Let S be a convex set and each f_i $(i = 1, \ldots, p)$ be a convex function on S. Then $x^\star \in S$ is a weak efficient solution of problem (13.3) if and only if there exist $\lambda_i \geq 0$ $(i = 1, \ldots, p)$, $\sum_{i=1}^{p} \lambda_i = 1$, such that $x^\star$ is an optimal solution of scalar problem*

$$\underset{x \in S}{Min} \sum_{i=1}^{p} \lambda_i f_i(x). \tag{13.12}$$

Theorem 13.4.4 *Let $x^\star \in S$ be an optimal solution of problem (13.4), for fixed $\lambda \in \Lambda$. Then $x^\star$ is an efficient solution of problem (13.3). Conversely, let S be a convex set and each f_i $(i = 1, \ldots, p)$ be a convex function on S. If $x^\star \in S$ is an efficient solution of problem (13.3) then there exist $\lambda_i \geq 0$ $(i = 1, \ldots, p)$, $\sum_{i=1}^{p} \lambda_i = 1$, such that $x^\star$ is an optimal solution of problem (13.12).*

Recall example 13.4.3, $x^\star = 0$ is an efficient solution of problem (13.10), and if we take $\lambda_1 = 0$, $\lambda_2 = 1$, then $x^\star$ is an optimal solution of the weighted scalar problem (13.12).

The thrust of the weighted sum approach (sometimes also refer as a *scalarization* of a multiobjective programming problem) is to convert a MOPP into a single objective programming problem, thereby making it convenient to handle the problem. The approach is simple and most widely used to solve MOPP. Of course there are some obvious difficulties in choosing the right proportion of weights. The weights reflect how much importance the decision maker wishes to give to various objective criteria. But in many situations it is hard to estimate the precise numerical values of the weights. For example, in an aircraft design problem, if one objective is to decrease the manufacturing cost of the aircraft and the other is to increase the safety of the passenger, it is difficult to assign the numeric values to the weights simply because human life is precious and can not be quantified. Consider another simple scenario where the government has to

provide more basic facilities to the weaker sections of the society at a low cost and at the same time wishes to decrease the subsidy on various goods. The conflict between the social responsibilities and economic development is difficult to be accurately measured by some weight vector. There is no mathematically designed mechanism to compute the weight vector, it completely rely on the knowledge of the decision maker, his preferences, and the set up of the problem. Furthermore, different weight vectors need not necessarily generate different properly efficient solutions of MOPP. For instance, in Example 13.4.1, $\lambda = \frac{1}{3}$ and $\lambda = \frac{2}{5}$ give the same optimal solution (3,5). This generally leads to wastage of search efforts in finding the properly efficient solutions of MOPP by weighted sum approach. The root cause behind this is that the mapping depicting the relationship between the weight vectors and the properly efficient solutions of MOPP is usually not known.

The practical limitations of the weighted sum approach motivate us to look for another approach which inherit the basic ideas of MOPP and at the same time computationally easy to implement. The search land us to 'goal programming'. We shall focus on this concept in the forthcoming sections.

13.5 Formulation of Goal Programming Problem

Rather than asking to provide the numerical value of weight for each objective criterion in MOPP (13.1), the decision maker is asked to rank the objectives according to their perceived importance. He is also asked to set up the aspired target values for each objective criterion. For instance, suppose $5x_1 + 2x_2 + 4x_3$ represents the profit function and $10x_1 + 3x_2 + 5x_3$ represents the maintenance cost of the finished goods inventory of some organization. An interactive discussion with the decision maker in an organization reveals that his first priority is to maximize profit and then to minimize the maintenance cost. Further, he also aspire to obtain a profit of Rs. 5000 and curtail the maintenance cost upto Rs. 1000. In this way, a dialogue with the decision maker reveals additional information of the aspired values for each objective criterion that can be used in formulation of a problem in a more constructive way.

The method of *goal programming* (GP) is based on this idea. It actually consists of formulating an optimization problem in such a manner that ensures that the objective criteria come close to the specified aspiration levels in order of priorities set up by the decision maker. It is worth to note that goal programming aims at satisfaction of the goals rather than exact achievement of the goals. Before proceeding further with our discussion we pause here to define related terminologies.

Definition 13.5.1 (Aspiration Level). *Aspiration level is the numerical value specified by the decision maker that reflects his desire or satisfactory level with regard to the objective function under consideration.*

Definition 13.5.2 (Goal). *An objective function along with its aspiration level is termed as goal.*

Definition 13.5.3 (Goal Deviation). *The difference between what we actually achieve and what we desire to achieve is called goal deviation. If the goal deviation is positive it reflects overachievement of a goal as the actual achieved value is more than the set up aspiration level. On the other hand negative value of the goal deviation indicates underachievement of the goal as the aspired level is more than what we could actually manage to achieve.*

For instance, suppose profit of an organization is described by a function $5x_1 + 2x_2 + 4x_3$. Now the decision maker wishes to attain a profit of at least Rs. 5000. So, the aspiration level of the decision maker is 5000 with regard to this objective function and the goal is described by an inequality $5x_1 + 2x_2 + 4x_3 \geq 5000$. If for some feasible vector $x^\star = (x_1^\star, x_2^\star, x_3^\star)$, $5x_1^\star + 2x_2^\star + 4x_3^\star > 5000$, then it shows that by taking the decision $x^\star$ the decision maker can achieve more profit than what he aspired. Whereas if for all feasible $x = (x_1, x_2, x_3)$, $5x_1 + 2x_2 + 4x_3 < 5000$, then no decision by the decision maker can help him to attain the aspired goal. This situation represents underachievement of the goal.

One obvious question arise here. How does the decision maker arrive at a figure of Rs. 5000? Arguably setting up too high value or too low value for the aspiration level is not advisable. First of all we assume that the decision maker is rationale and knowledgeable. Secondly, in a MOPP (13.1) with p objective criteria, one can solve p individual scalar programming problems

$$\underset{x \in S}{Min}\ f_i(x) \qquad (i = 1, \ldots, p). \tag{13.13}$$

Each of these problems, being single objective nonlinear constrained programming problem, can be solved by the techniques described in earlier chapters. Once the optimal values, say $f_i(x^{(i)\star})$ $(i = 1, \ldots, p)$, are known, they can be used as aspiration levels for the respective objective functions.

We now turn our attention to formulate a mathematical model of goal programming problem.

Suppose the i-th objective function is described by $f_i(x)$, $x \in \mathbf{R}^n$, and its aspiration level is specified by $v_i \in \mathbf{R}$. The possible form of the i-th goal is

$$\begin{aligned} \text{either} \quad & f_i(x) \leq v_i, \\ \text{or} \quad & f_i(x) \geq v_i, \\ \text{or} \quad & f_i(x) = v_i. \end{aligned}$$

By introducing two additional variables, $d_i^- \geq 0$, $d_i^+ \geq 0$, we can transform the above relations into the following equation

$$f_i(x) + d_i^- - d_i^+ = v_i. \tag{13.14}$$

It is important to note here that if the goal is of the form $f_i(x) \leq v_i$, then d_i^+ is the undesirable variable in equation (13.14). Because if $d_i^- = 0$ and $d_i^+ > 0$ then equation (13.14) implies $f_i(x) = v_i + d_i^+ > v_i$, thereby not satisfying the set up goal $f_i(x) \leq v_i$. While, if $d_i^- > 0$ and $d_i^+ = 0$, then we get $f_i(x) = v_i - d_i^- < v_i$, consequently, achieving the set up goal. Thus, $f_i(x) \leq v_i$ implies d_i^+ is an undesirable variable and we need to minimize it. Similarly, $f_i(x) \geq v_i$, gives d_i^- as an undesirable variable, and $f_i(x) = v_i$ yields $d_i^- + d_i^+$ as an undesirable expression. In all situations, we first identify the undesirable variable/expression in the goal and then attempt to minimize the same. The variable d_i^- is called negative deviational variable while d_i^+ is called the positive deviational variable for obvious reasons.

Recall the MOPP

$$\underset{x \in S}{Min}\ f(x) = (f_1(x), \ldots, f_p(x)). \tag{13.15}$$

Suppose the feasible set S is described by the m inequality constraints $g_i(x) \leq 0$ $(i = 1, \ldots, m)$. Some of the constraints could be $\geq$ or $=$ types.

Since the form of the goal function and the constraint function are alike, both are in the form of mathematical inequalities or equations, goal programming treats the constraint functions also as goals. Thereby the constraints too are converted into equations using negative and positive deviational variables. In other words, constraint $g_i(x) \leq 0$ is expressed as $g_i(x) + d_i^- - d_i^+ = 0,\ d_i^-, d_i^+ \geq 0$.

To solve an optimization problem, we must ensure feasibility of the problem. This is accomplished by assigning first priority to the goals representing the actual constraint functions. Once we attain this priority, we have a feasible solution of the problem in hand. The actual constraints are therefore termed as *rigid goals or hard goals* whereas the goals representing the original objective functions are called *soft goals*.

Summarizing the above discussion we conclude the following procedure for the formulation of the goal programming problem (GPP).

1. Specify the aspiration level for each objective function.
2. Set up the goals and convert them into equations by using negative and positive deviational variables.
3. Treat the constraints present in the problem as goals and convert them into equations by using negative and positive deviational variables.
4. Assign first priority to the hard constraints and rank all other goals according to their importance specified by the decision maker.
5. Identify the appropriate undesirable deviational variables or expressions involving deviational variables and make an attempt to minimize them in order of the specified priorities.
6. The task in point 5 is accomplished by constructing a suitable multiobjective programming problem and by defining an appropriate ranking mechanism.

The following example illustrates the above procedure.

Example 13.5.1 *Formulate the following three objective LPP as a goal programming problem.*

$$\begin{aligned} \textit{Min} \quad & (2x_1 - x_2,\ 4x_1 - 5x_2,\ -x_1) \\ \textit{subject to} \quad & \\ & 4x_1 + 5x_2 \leq 20 \\ & 3x_1 + 2x_2 \leq 12 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Solution Suppose the decision maker give first priority to the second objective criterion, second priority to the third objective criterion and third priority to the first objective criterion. He wish to keep the first, second and third priorities objective values below $8, -2, 1$, respectively. So, the goals in order of their importance are given by

$$\begin{aligned} 4x_1 - 5x_2 &\leq 8 \\ x_1 &\geq 2 \\ 2x_1 - x_2 &\leq 1. \end{aligned}$$

Introducing the deviational variables both in the actual goals and the two constraints (hard goals), we get the following system of linear equations

$$\begin{aligned} 4x_1 + 5x_2 + d_1^- - d_1^+ &= 20 \\ 3x_1 + 2x_2 + d_2^- - d_2^+ &= 12 \\ 4x_1 - 5x_2 + d_3^- - d_3^+ &= 8 \\ x_1 + d_4^- - d_4^+ &= 2 \\ 2x_1 - x_2 + d_5^- - d_5^+ &= 1 \\ x_j,\ d_i^-,\ d_i^+ &\geq 0,\ (j = 1, 2)\ (i = 1, \ldots, 5). \end{aligned} \tag{13.16}$$

The undesirable variables are, $d_1^+,\ d_2^+,\ d_3^+,\ d_4^-,\ d_5^+$, respectively.

Our next task is to construct an objective function depicting the priorities of the decision maker. The first rank is reserved for the hard constraints followed by the second objective function, third objective function and then the first objective function, respectively. Therefore, we make an attempt to minimize $(d_1^+ + d_2^+,\ d_3^+,\ d_4^-,\ d_5^+)$ in this order only. Thus the GPP becomes

$$\begin{aligned} \text{Min} \quad & (d_1^+ + d_2^+,\ d_3^+,\ d_4^-,\ d_5^+) \\ \text{subject to} \quad & (13.16). \end{aligned}$$

Note that we have yet to specify the meaning of 'minimization in order'.

Remark 13.5.1 *GPP is a MOPP in which the objective criteria in terms of the deviational variables are ranked according to their importance and the minimization process has to take care of the preassigned order. We simply can not bypass this order. Obviously this ranking is different from the concepts of weak efficiency, efficiency or proper efficiency, as in the latter definitions the order of the objective criterion is immaterial.*

Definition 13.5.4 (Lexicographic Minimum Vector). *A vector $w^{(1)} \in \mathbf{R}^p_+$ is said to be preferred to another vector $w^{(2)} \in \mathbf{R}^p_+$ if $w^{(1)}_i = w^{(2)}_i$ $(i = 1, \ldots, k-1)$ and $w^{(1)}_k < w^{(2)}_k$, for some k $(k = 1, \ldots, p)$. For example, if $w^{(1)} = (0, 12, 3, 17, 25)$ and $w^{(2)} = (0, 12, 5, 11, 27)$, then $w^{(1)}$ is preferred to $w^{(2)}$. If, in some set $\mathfrak{B} \subseteq \mathbf{R}^p_+$, there is no vector preferred to $w^\star \in \mathfrak{B}$ then $w^\star$ is called the lexicographic minimum vector in the set $\mathfrak{B}$.*

It is worth to note here that in choosing the lexicographic minimum among the possible candidate vectors, we search a vector with minimum first component. If this choice is unique, we stop, else we continue to choose the minimum second component without altering the minimum first component. Repeat this procedure till we get a unique vector with first k minimum components, k can be equal to p also. The procedure is somewhat similar to searching an english word in a dictionary. The minimization in GPP is taken in the sense of lexicographic minimum (Lexi-Min).

Thus a general model of GPP is described as follows

$$\text{Lexi-Min} \quad (F_1(d^-, d^+), \ldots, F_K(d^-, d^+))$$

subject to

$$\begin{aligned} f_i(x) + d_i^- - d_i^+ &= v_i \quad (i = 1, \ldots, p) \\ g_j(x) + d_j^- - d_j^+ &= 0 \quad (j = 1, \ldots, m) \\ d_i^-, d_j^-, d_i^+, d_j^+ &\geq 0 \quad (i = 1, \ldots, p)\ (j = 1, \ldots, m), \end{aligned} \tag{13.17}$$

where K is the number of priorities specified by the decision maker. Remember the first priority is reserved for the hard goals. Also, $F_r(d^-, d^+)$ $(r = 1, \ldots, K)$, are linear functions of the deviational variables.

If all the functions f_i and g_j are linear functions of the decision variable x then the GPP (13.17) is called *linear goal programming problem* (LGPP). In the next section we discuss the solution methodologies to solve LGPPs.

13.6 Solution Methodologies for Linear Goal Programming Problems

We briefly present two traditionally known techniques to solve LGPPs. If the decision variable x involved in LGPP belongs to $\mathbf{R}^2$ then the problem can be solved by graphical technique. We illustrate this technique on Example 13.5.1. For convenience, we reintroduce the example.

Example 13.6.1 *Solve* *Lexi-Min* $(d_1^+ + d_2^+,\ d_3^+,\ d_4^-,\ d_5^+)$

subject to

$$\begin{aligned}
G_1: \quad & 4x_1 + 5x_2 + d_1^- - d_1^+ = 20 \\
G_2: \quad & 3x_1 + 2x_2 + d_2^- - d_2^+ = 12 \\
G_3: \quad & 4x_1 - 5x_2 + d_3^- - d_3^+ = 8 \\
G_4: \quad & x_1 + d_4^- - d_4^+ = 2 \\
G_5: \quad & 2x_1 - x_2 + d_5^- - d_5^+ = 1 \\
& x_j,\ d_i^-,\ d_i^+ \geq 0 \ (j = 1,2) \quad (i = 1,\ldots,5).
\end{aligned} \tag{13.18}$$

Solution Ignoring the deviational variables, the five goals are plotted as straight lines in Fig 13.9.

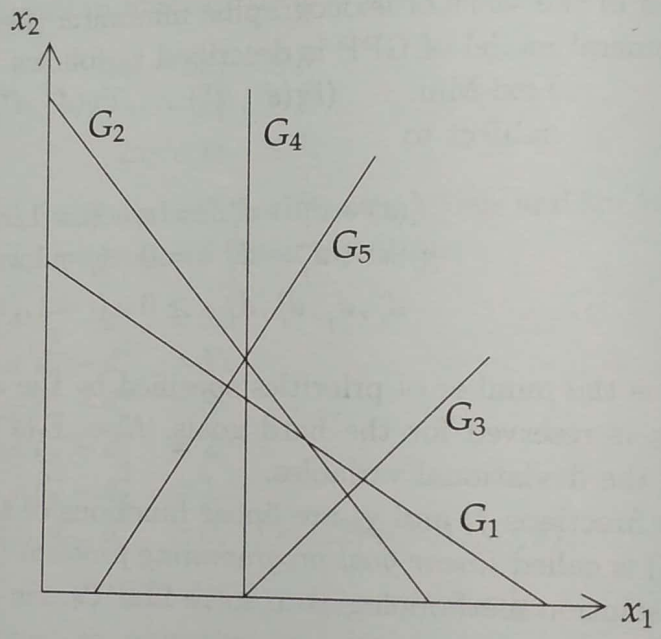


Fig. 13.9.

Concentrate on the first objective function, Min $(d_1^+ + d_2^+)$. The corresponding goals are G_1, G_2. We identify the region in the first figure in Fig 13.10 depicting the optimal solutions of the first priority objective criteria. Observe that in the shaded region $d_i^+ = 0$ $(i = 1,2)$, thereby yielding zero optimal value of the objective function.

Next, move to minimize the second objective d_3^+, with corresponding goal G_3, over the optimal solution space of the first objective function (i.e. the shaded region in the first figure in Fig 13.10). Optimal value of this objective is zero and the new optimal solution space is shown in the second graph in Fig 13.10. This region will act as the feasible region for the third objective function of the GPP.

We repeat the above procedure with the third objective to get its optimal value zero in the new optimal solution space shown in Fig 13.11. This is the feasible region for the successive objective function.

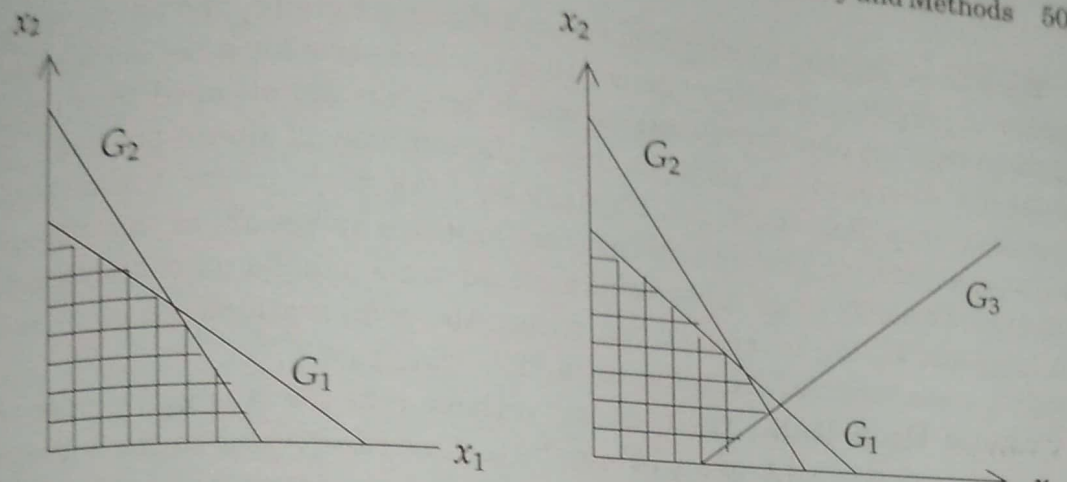


Fig. 13.10.

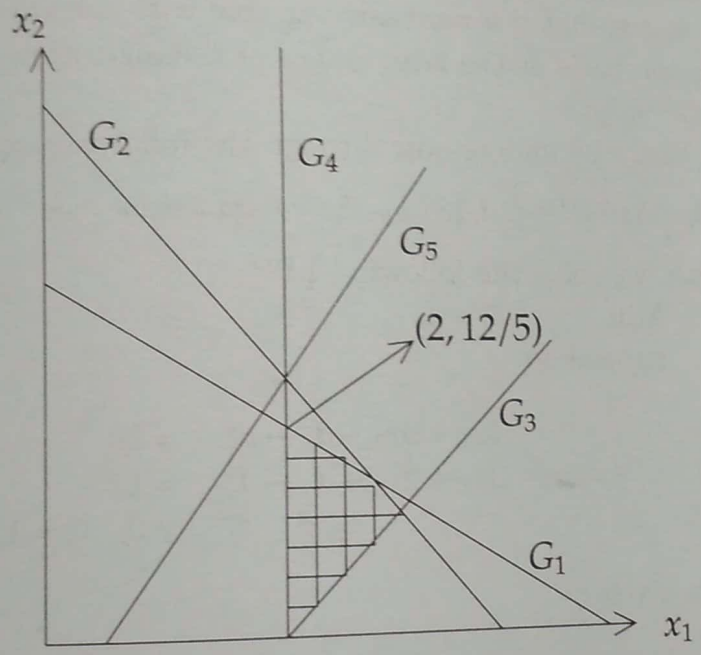


Fig. 13.11.

The final priority goal G_5 is outside the current feasible region, we move the objective line G_5 parallel to itself in the upward direction to meet the feasible region. The lexicographic optimal solution is $x^\star = (2, 2.4)$ and the optimal achievement value of the objective function of (13.17) is $(0, 0, 0, 0.6)$. The positive value 0.6 in the end indicates that the aspired value 'below 1' for the objective criterion $2x_1 - x_2$ can not be achieved. Infact we can not bring down its value below 1.6. But since this objective was at least prior to the decision maker, so, the compromise is reasonably acceptable.

In higher dimension set up we can not rely on the graphical technique. So, we switch over to the *sequential goal programming technique* wherein the ideas of the graphical technique are preserved.

The scheme involves solving K number of linear programs, one by one, where K is the number of priorities, in such a manner that the lexicographic order is maintained and at the same time the objective functions which have already attained their optimal values should not deteriorate in their values in the subsequent linear programs. The latter is ensured by adding additional constraints, $f_i(x) = a_i^\star$, where f_i are those objective functions which have already achieved their optimal values $a_i^\star$, in the subsequent linear programs. This may cause substantial increase in the number of constraints particulary in a large size GPPs. Luckily we can reduce the computational effort by progressively dropping some variables according to the following rule.

Column Drop Rule. Any nonbasic variable that has a negative opportunity cost $z_j - c_j$ in the optimal table of a LPP can be assigned zero value in the subsequent linear programming problems and therefore the column corresponding to this variable can be dropped from the subsequent linear programming problems.

Implicitly the rule states that if a nonbasic variable with negative opportunity cost $z_j - c_j$ is introduced in the basis at the later stages of the algorithm it will degrade the solution in the Lexi-Min order.

The algorithm can easily be understood through the following example.

Example 13.6.2 *Solve the GPP (13.18) by the column drop rule.*

Solution To begin with, we solve the following LPP

$$\begin{aligned} &\text{Min} && d_1^+ + d_2^+ \\ &\text{subject to} \\ & && 4x_1 + 5x_2 + d_1^- - d_1^+ && = 20 \\ & && 3x_1 + 2x_2 + d_2^- - d_2^+ && = 12 \\ & && x_i, d_i^-, d_i^+ && \geq 0 \quad (i = 1, 2). \end{aligned}$$

The optimal table is given by

	x_1	x_2	d_1^+	d_2^+
$d_1^- = 20$	4	5	-1	0
$d_2^- = 12$	3	2	0	-1
$z_j - c_j$	0	0	-1	-1
			✓	✓

'✓' denotes that these non basic variables have negative $z_j - c_j$, hence according to the column drop rule, the marked columns can be dropped from the subsequent iterations. Observe that the optimal value of the LPP is zero.

The next level LPP and its optimal solution are respectively given by

Min d_3^+
subject to

$$\begin{aligned} 4x_1 + 5x_2 + d_1^- &= 20 \\ 3x_1 + 2x_2 + d_2^- &= 12 \\ 4x_1 - 5x_2 + d_3^- - d_3^+ &= 8 \\ x_j, d_i^-, d_i^+ &\geq 0, \quad (j = 1,2)\ (i = 1,2,3). \end{aligned}$$

	x_1	x_2	d_3^+
$d_1^- = 20$	4	5	0
$d_2^- = 12$	3	2	0
$d_3^- = 8$	4	-5	-1
$z_j - c_j$	0	0	-1
			✓

The optimal value is zero. We then move on to construct the LPP corresponding to the third priority objective function.

Min d_4^-
subject to

$$\begin{aligned} 4x_1 + 5x_2 + d_1^- &= 20 \\ 3x_1 + 2x_2 + d_2^- &= 12 \\ 4x_1 - 5x_2 + d_3^- &= 8 \\ x_1 + d_4^- - d_4^+ &= 2 \\ x_j, d_i^-, d_i^+ &\geq 0 \quad (j = 1,2)\ (i = 1,\ldots,4). \end{aligned}$$

The optimal table is shown below.

	x_2	d_4^-	d_4^+
$d_1^- = 12$	5	-4	4
$d_2^- = 6$	2	-3	3
$d_3^- = 0$	-5	-4	4
$x_1 = 2$	0	1	-1
$z_j - c_j$	0	-1	0
		✓	

The final priority criterion is represented by the following LPP, and its optimal solution table is subsequently generated.

$$\text{Min} \quad d_5^+$$

subject to

$$\begin{aligned} 4x_1 + 5x_2 + d_1^- &= 20 \\ 3x_1 + 2x_2 + d_2^- &= 12 \\ 4x_1 - 5x_2 + d_3^- &= 8 \\ x_1 - d_4^+ &= 2 \\ 2x_1 - x_2 + d_5^- - d_5^+ &= 1 \\ x_j, d_i^-, d_i^+ &\geq 0 \quad (j = 1,2)\ (i = 1,\ldots,5). \end{aligned}$$

The optimal table is shown below. Notice that the optimal value is zero.

	d_1^-	d_5^-	d_4^+
$x_2 = 12/5$	1/5	0	4/5
$d_2^- = 6/5$	-2/7	0	7/5
$d_3^- = 12$	1	0	8
$d_5^+ = 3/5$	-1/5	-1	-14/5
$x_1 = 2$	0	0	-1
$z_j - c_j$	-1/5	-1	-14/5

The optimal value of the LPP is 3/5. Thus the lexicographic optimal value of the GPP is $(0, 0, 0, 0.6)$.

The optimal solution of the three objective LPP in the Example 13.5.1 is $x_1^\star = 2$, $x_2^\star = 2.4$ and the objective value in order of importance is $(-0.4, -2, 1.6)$.

13.7 Summary and Additional Notes

- Section 13.2 provides glimpses of multiobjective programming by briefly discussing the basic issue of trade-off between the objective criteria.
- Section 13.3 describes various solutions concepts related to the multiobjective programming problems. The concepts are well illustrated through many examples. For more reading refer to the texts by Jahn [82] and Sawaragi et al. [140].
- The most commonly used technique to solve a MOPP is to use scalarization and convert MOPP into a scalar programming problem. This approach is explained in Section 13.4. The best reading is undoubtably the research article by Geoffrion [67].
- Section 13.5 is devoted to describe a goal programming formulation of a MOPP. This is another classical approach to solve MOPP. The graphical and the sequential goal programming techniques for linear goal programming problems are illustrated in section 13.6. The text books by Ignizio [81], Hillier and Liebermann [76] and the survey article by Aouni and Kettani [4] contain good material on goal programming.

- Numerous research articles are devoted to study the KKT type optimality conditions and the duality results for nonlinear MOPP. We cite here only those contributions which had been novel in their approach, like, the works of P. Wolfe [167], Weir and Mond [163], Chankong and Haimes [33], Charnes and Cooper [34], Egudo [52], Jeyakumar and Mond [85], Singh [144], Hanson [75], to name a few. While citing the above references we have restricted ourself to differentiable settings. One can obviously go deep to find many interesting results for MOPP in nonsmooth versions too.
- *Evolutionary algorithms* have been successfully used to solve MOPP's. The primary reason for their success is the ability of the evolutionary algorithms to generate several solutions in a single simulation run. A brief description of few such algorithms is given later in this book. The beginners in the field of multiobjective optimization with evolutionary algorithms can refer to a good text by Deb [46].
- Extensive research in last many decades resulted in various techniques, like, multiattribute utility analysis, normal boundary intersection method, multiobjective heuristic algorithms, among others, for computing efficient frontiers of extremely complex engineering design problems and decision making problems. However, none of these techniques are perfect and selecting among them depends on the requirements of the problem. For more details on the solution methodologies, we suggest a website dedicated to the multiobjective programming: http://www.lania.mx/~ccoello/EMOO.
- In this Chapter we have focussed on MOPP in finite dimensional real space. However, vector optimization problems set up in very abstract spaces have been investigated in detail in last many years. Many excellent texts are available to get the insight of the otherwise extremely involved subject. For instance, one can refer to, Borwein [25], Jahn [82], Luc [105], Luenberger [106], Steuer [148], Sawaragi et al. [140]. Furthermore, numerous research articles can be found dealing with nonsmooth nonlinear MOPPs.

13.8 Exercises

13.1 *Consider the linear MOPP*

$$\begin{aligned} &\textit{Max} && f(x) = \{x_1 + 5x_2,\ x_1\} \\ &\textit{subject to} \\ & && x_1 - 2x_2 \leq 2 \\ & && x_1 + 2x_2 \leq 12 \\ & && 2x_1 + x_2 \leq 9 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

(i) Show graphically that the optimal solutions taken separately for two linear programming problems, each with a single objective function, do not coincide.

(ii) Determine graphically whether each of the given feasible solution is an efficient solution of the problem: $(2,0)$, $(4,7)$, $(3,3)$, $(2,5)$, $(2,2)$, $(0,6)$.
(iii) Identify the efficient frontier of this model in an objective values space.

13.2 *Do the above exercise for the linear MOPP*

$$\begin{aligned} &\textit{Min} && \{5x_1 - x_2,\ x_1 + 4x_2\} \\ &\textit{subject to} \end{aligned}$$

$$\begin{aligned} -5x_1 + 2x_2 &\le 10 \\ x_1 + x_2 &\ge 3 \\ x_1 + 2x_2 &\ge 4 \\ x_1,\ x_2 &\ge 0, \end{aligned}$$

and points $(4,0)$, $(2,1)$, $(3,3)$, $(1,2)$, $(5,0)$, $(0,0)$.

13.3 *Let* $S = \{(x_1, x_2, x_3) \in \mathbf{R}^3 : x_1^2 \le x_2^3\}$. *Define a function* $f : S \to \mathbf{R}^2$ *by*

$$f(x_1, x_2, x_3) = (x_3^2 - x_1,\ x_2^2 - \sqrt[3]{x_3^7}).$$

Is $(0,0,0)$ *an efficient solution of the MOPP:* $\underset{x \in S}{\textit{Min}}\ f(x)$*? Give reasons for the answer.*

13.4 *Let* $S = [0,2]$. *Define* $f : S \to \mathbf{R}^2$ *by*

$$f(x) = \begin{cases} (x - 1, 1 - x), & x \in [0,1] \\ (0, 1 - x), & x \in [1,2]. \end{cases}$$

Find all the weak efficient solutions and efficient solutions of the two objective programming problem $\underset{x \in S}{\textit{Min}}\ f(x)$.

13.5 *Using the weighted sum approach, determine the efficient frontier of the following linear multiobjective programming problems.*

(i)

$$\begin{aligned} &\textit{Max} && \{x_1 + 2x_2,\ -2x_1 - 4x_2\} \\ &\textit{subject to} \end{aligned}$$

$$\begin{aligned} -x_1 + 2x_2 &\le 4 \\ x_1,\ x_2 &\ge 0. \end{aligned}$$

(ii)

$$\begin{aligned} &\textit{Max} && \{-4x_1 + x_2,\ x_1 - x_2\} \\ &\textit{subject to} \end{aligned}$$

$$\begin{aligned} -x_1 + 2x_2 &\le 8 \\ -x_1 + 2x_2 &\ge 4 \\ x_1,\ x_2 &\ge 0. \end{aligned}$$

(iii) *Max* $\{4x_1 + 2x_2,\ 8x_1 + 10x_2\}$
subject to

$$\begin{aligned} x_1 + x_2 &\le 70 \\ x_1 + 2x_2 &\le 100 \\ x_1 &\le 60 \\ x_2 &\le 40 \\ x_1, x_2 &\ge 0. \end{aligned}$$

13.6 *Using the weighted sum approach, find the efficient frontier of the following multiobjective nonlinear programming problems.*

(i) *Min* $\{2x_1x_2,\ x_1^2 + x_2^2\}$;
$x \in \mathbf{R}^2$

(ii) *Min* $\{x_1^2 + x_2^2,\ 3 - x_1 + x_2^2\}$
subject to $-3 \le x_1, x_2 \le 3.$

13.7 *Consider the following MOPP*

Min $\{-x_1 + x_2,\ x_1^2 + x_2^2\}$
subject to

$$\begin{aligned} x_1^2 + x_2^2 &\le 2 \\ -x_1 + x_2^2 &\ge 1 \\ x_1 - x_2 &\le 1 \\ x_1, -x_2 &\ge 0. \end{aligned}$$

Find the efficient frontier of the problem.

13.8 *Consider the following MOPP*

Max $\{x_1,\ -x_1 - x_2^2\}$
subject to

$$\begin{aligned} x_1^2 - x_2 &\le 0 \\ x_1 + 2x_2 &\le 3. \end{aligned}$$

Determine all the efficient solutions of the problem corresponding to the weight vectors $(\frac{1}{2}, \frac{1}{2})$ *and* $(\frac{1}{3}, \frac{2}{3})$. *Is any of the generated efficient solution a properly efficient solution of the problem? Justify your answer.*

13.9 *Determine* $x = (x_1, x_2)$ *so as to*

(i) *Lexi - Min* $\{(d_1^+ + d_2^+),\ (d_3^-),\ (d_4^+),\ (d_5^+)\}$
subject to

$$\begin{aligned} 4x_1 + 5x_2 + d_1^- - d_1^+ &= 80 \\ 4x_1 + 2x_2 + d_2^- - d_2^+ &= 48 \\ 80x_1 + 100x_2 + d_3^- - d_3^+ &= 800 \\ x_1 + d_4^- - d_4^+ &= 6 \\ x_1 + x_2 + d_5^- - d_5^+ &= 7 \\ x, d^-, d^+ &\ge 0. \end{aligned}$$

(ii) *Lexi - Min* $\{(d_1^-), (d_3^-), (d_2^-), (d_1^+ + d_2^+)\}$

subject to

$$\begin{aligned} 2x_1 + x_2 + d_1^- - d_1^+ &= 20 \\ x_1 + d_2^- - d_2^+ &= 12 \\ x_2 + d_3^- - d_3^+ &= 10 \\ x, d^-, d^+ &\geq 0. \end{aligned}$$

(iii) *Lexi - Min* $\{(d_1^- + d_1^+), (2d_2^+ + d_3^+)\}$

subject to

$$\begin{aligned} x_1 - 10x_2 + d_1^- - d_1^+ &= 50 \\ 3x_1 + 5x_2 + d_2^- - d_2^+ &= 20 \\ 8x_1 + 6x_2 + d_3^- - d_3^+ &= 100 \\ x, d^-, d^+ &\geq 0. \end{aligned}$$

(iv) *Lexi - Min* $\{(d_1^+, d_2^-, d_3^-)\}$

subject to

$$\begin{aligned} x_1 - x_2 + d_1^- - d_1^+ &= 10 \\ 2x_1 + x_2 + d_2^- - d_2^+ &= 26 \\ -x_1 + 2x_2 + d_3^- - d_3^+ &= 6 \\ x, d^-, d^+ &\geq 0. \end{aligned}$$

14 Semi-definite Programming

14.1 Introduction

In this chapter, we focus on a special class of nonlinear programming problems called semi-definite programming problems (SDPPs). This class can be considered as an extension of the class of LPPs wherein the data in the problem comprises of matrices. The constraints in such optimization problems involve comparison of two matrices over the set of symmetric matrices. For this purpose, the set of symmetric matrices is equipped with the positive semi-definite partial order relation. The resultant partially ordered set of symmetric positive semi-definite matrices possesses many promising properties that eventually lead to efficient solution procedures for the class of semi-definite programming problems.

14.2 Motivation

Before going into the technicalities of the main problem we lay the foundation of the subject and get familiar with it. While passing from the linear programming problems to the nonlinear programming problems, it was very natural for us to allow one or more than one function appearing in the linear programming problem (LPP) to be nonlinear function of the decision variable x. This kind of nonlinear programming problem model is described by

$$\begin{aligned} &\text{Min} && f(x) \\ &\text{subject to} \\ & && g_i(x) \geq 0 \quad (i = 1, \ldots, m), \end{aligned} \tag{14.1}$$

and it has been a subject of study in previous chapters. There is another way to introduce nonlinearity in LPP model. Although it may not occur very naturally to us yet by retaining linearity in the objective function and the constraint functions one can introduce nonlinearity in the problem by making the *order relation* '$\geq$' *as nonlinear.* We explain the meaning of this statement through some examples.

Consider the nonlinear optimization problem

$$\begin{aligned} &\text{Min} && x_1 \\ &\text{subject to} \\ & && 4x_1x_2 \geq 1 \\ & && x_1 + x_2 \geq 0. \end{aligned}$$

The problem can be equivalently expressed as

$$\begin{aligned} &\text{Min} && x_1 \\ &\text{subject to} \\ & && \sqrt{(x_1 - x_2)^2 + 1} \leq x_1 + x_2, \end{aligned}$$

i.e.

$$\begin{aligned} &\text{Min} && x_1 \\ &\text{subject to} \\ & && (x_1 - x_2,\ 1,\ x_1 + x_2) \in S, \end{aligned}$$

where $S = \left\{(v_1, v_2, v_3) \in \mathbf{R}^3 : v_3 \geq \sqrt{v_1^2 + v_2^2}\right\}$.

If we consider the problem

$$\begin{aligned} &\text{Min} && x_2 - x_1 \\ &\text{subject to} \\ & && x_1^2 + x_2^2 \leq 2 \\ & && x_2^2 - x_1 \leq 0, \end{aligned}$$

then after simple calculations it can be seen that the problem is equivalent to

$$\begin{aligned} &\text{Min} && x_2 - x_1 \\ &\text{subject to} \\ & && \begin{pmatrix} 1 & 0 & x_1 \\ 0 & 1 & x_2 \\ x_1 & x_2 & 2 \end{pmatrix} \in S_1 \\ & && \begin{pmatrix} 1 & x_2 \\ x_2 & x_1 \end{pmatrix} \in S_2, \end{aligned}$$

where S_1 and S_2 are, respectively, the sets of 3×3 and 2×2 symmetric positive semi-definite matrices.

Notice that in the above illustrations the nonlinear constraints are converted into linear constraints over appropriate sets. The resulting equivalent problems thereby preserve linear structure in both the objective function as well as in the constraint expressions and bring in the nonlinearity in the form of set in the constraints. At this stage, one

may wonder as to how to identify an appropriate set over which a nonlinear constraint function can be converted into a linear constraint function? The answer will get unfolded during the course of our discussion. For the time being we assume that we have the desired skill to convert a nonlinear inequality into a linear inequality over an appropriate set. What we would like to stress here upon is that the considered problems are nonlinear programming problems in classical sense but the corresponding equivalent problems are virtually LPPs over some set. If we relook at the usual LPP

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && Ax \geq b \\ & && x \geq 0, \end{aligned} \tag{14.2}$$

it can be reframed as

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && Ax - b \in \mathbf{R}^m_+ \\ & && -x \in \mathbf{R}^n_+. \end{aligned} \tag{14.3}$$

In other words, it is same as asking to solve

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && \hat{A}x - \hat{b} \in S, \end{aligned}$$

where $\hat{A} = \begin{pmatrix} A \\ -I \end{pmatrix}$, $\hat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}$, and $S = \mathbf{R}^m_+ \times \mathbf{R}^n_+$.

Thus, one can say that the given LPP is a linear programming problem over the set S. The equivalent problems formulated in the earlier explained examples are similar to LPPs but over the different sets. In case of the usual LPP, the set is the product of nonnegative orthants $\mathbf{R}^m_+ \times \mathbf{R}^n_+$ whereas in the nonlinear cases described above, the sets are, $\left\{(v_1, v_2, v_3) : v_3 \geq \sqrt{v_1^2 + v_2^2}\right\}$, geometrically an ice-cream cone in three dimensional Euclidean space, or $S_1 \times S_2$, the product of the sets of positive semi-definite matrices. We therefore conclude that the basic difference between the usual LPP and the above cited nonlinear programming problems lies in the structure of the sets involved in the constraints.

Through the above discussion we tried to explain that by preserving linearity in the objective and the constraint functions and bringing in nonlinearity in the form of a set S, i.e. in the order relation '$\geq$', one can build a new model of nonlinear optimization problem which actually do not involve nonlinear functions.

Having said this, there obviously emerge few potential questions. Can any set S used to describe such models be useful? Does S require some kind of meaningful structure which in turn helps to develop a solution methodology for such class of optimization problems? Is this kind of modeling interesting purely from the theoretical point of view? Can problems arising in practice be casted into such models? What are the existing theoretical and computational results in this direction? We shall be addressing to these questions in the sections to follow.

14.3 Cone

Observe that the inequality $Ax \geq b$ in (14.2) has been expressed as $Ax - b \in \mathbf{R}^m_+$, in (14.3), $\mathbf{R}^m_+$ is the nonnegative orthant of $\mathbf{R}^m$. What is special about $\mathbf{R}^m_+$? Let us focus on this issue.

The inequality '$\geq$' in $\mathbf{R}^m$ means that for $a, b \in \mathbf{R}^m$,

$$a \geq b \Leftrightarrow a - b \in \mathbf{R}^m_+ \Leftrightarrow a_i \geq b_i \quad (i = 1, \ldots, m).$$

Thus '$\geq$' is the coordinate-wise ordering of vectors in $\mathbf{R}^m$ that satisfies number of basic properties.

(P1) *Reflexive* : $a \geq a$, $\forall\, a \in \mathbf{R}^m$.
(P2) *Antisymmetric* : $a \geq b$ and $b \geq a \Rightarrow a = b$, $\forall\, a, b \in \mathbf{R}^m$.
(P3) *Transitive* : $a \geq b$ and $b \geq c \Rightarrow a \geq c$, $\forall\, a, b, c \in \mathbf{R}^m$.
(P4) *Homogenous* : $a \geq b$ and $\lambda \geq 0 \Rightarrow \lambda a \geq \lambda c$, $\forall\, a, b \in \mathbf{R}^m$, $\lambda \in \mathbf{R}$.
(P5) *Additive* : $a \geq b$ and $c \geq d \Rightarrow a + c \geq b + d$, $\forall\, a, b, c, d \in \mathbf{R}^m$.

The above properties make '$\geq$' a partial order relation on the set $\mathbf{R}^m$ which is compatible with linear operations. We hardly notice this while studying LPP yet it is these very properties that imparted nice geometric features to LPP. These features are implicitly utilized in the previous chapters and we need not repeat them. But at this moment we would certainly like to ask 'is '$\geq$' the only possible partial order that can define the vector inequality satisfying the above five properties?' The quest to answer this question leads us to the notion of *cone*.

Definition 14.3.1 (Cone). *A non empty subset K of a vector space X is called a cone if, $k \in K$, $\lambda \in \mathbf{R}$, $\lambda \geq 0 \Rightarrow \lambda k \in K$.*

For example, consider the sets

(*i*) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 = 0\}$;
(*ii*) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq |x_1|\}$;
(*iii*) $K = \{(x_1, x_2, x_3) \in \mathbf{R}^3 : x_1 > 0, x_2 > 0, x_3 > 0\} \cup \{(x_1, x_2, x_3) \in \mathbf{R}^3 : x_1 \geq x_2 \geq 0, x_3 = 0\}$;
(*iv*) $K = \{A \in \mathbf{R}^{n \times n} : A^T = A,\ x^T A x \geq 0,\ \forall\, x \in \mathbf{R}^n\}$;
(*v*) $K = \{x = \{x_n\} : x_n = 0$ for all but finitely many n$\}$.

It can easily be verified that all the above described sets are infact cones.

Definition 14.3.2 (Convex Cone). *A cone K is said to be a convex cone if K is a convex set.*

Remark 14.3.1 *It is obvious that K is a convex cone if and only if for all $k_1, k_2 \in K$, $k_1 + k_2 \in K$. In other words, K is closed under addition.*

The cones listed above are convex cones. However, the following cones are not convex cones.

$$\begin{aligned}
&(i)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 = 0\} \cup \{(x_1, x_2) \in \mathbf{R}^2 : x_2 = 0\};\\
&(ii)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : |x_1| \leq |x_2|\};\\
&(iii)\ K = \{(x_1, x_2, x_3) \in \mathbf{R}^3 : x_1 \geq x_2 \geq 0,\ x_3 = 0\} \cup \{(x_1, x_2, x_3) \in \mathbf{R}^3 : x_3 \geq x_1 \geq 0,\ x_2 = 0\}.
\end{aligned}$$

Definition 14.3.3 (Pointed Cone). *A cone K is said to be a pointed cone if $K \cap (-K) = \{0\}$, i.e. if $k \in K$ and $-k \in K$ then $k = 0$.*

For instance,

$$\begin{aligned}
&(i)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq |x_1|\};\\
&(ii)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : |x_1| \leq |x_2|\};\\
&(iii)\ K = \{A \in \mathbf{R}^{n \times n} : A^T = A,\ x^T A x \geq 0,\ \forall\, x \in \mathbf{R}^n\};
\end{aligned}$$

are pointed cones while

$$\begin{aligned}
&(i)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 = 0\};\\
&(ii)\ K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 \geq 0\} \cup \{(x_1, x_2) \in \mathbf{R}^2 : x_1 \leq 0,\ x_2 \geq 0\};\\
&(iii)\ K = \{x = \{x_n\} : x_n = 0 \text{ for all but finitely many n}\};
\end{aligned}$$

are not pointed cones. Geometrically, a pointed cone should not contain a hyperplane passing through the origin.

Let K be a non empty pointed convex cone in $\mathbf{R}^m$. We induce a new order in $\mathbf{R}^m$ by

$$a \geq_K b \quad \Leftrightarrow \quad a - b \in K.$$

This order relation satisfies the five properties $(P1)$ to $(P5)$. The property $(P1)$, reflexivity, follows as $0 \in K$ by definition of cone, and $(P2)$, antisymmetry, follows on account of K being pointed, while $(P3)$, transitivity, holds in lieu of convexity of K as

$$a \geq_K b \text{ and } b \geq_K c \Leftrightarrow a - b \in K,\ b - c \in K \Leftrightarrow (a - b) + (b - c) \in K \Leftrightarrow a \geq_K c.$$

Further, the property $(P4)$, homogeneity, follows from the definition of cone and $(P5)$, additivity, is due to convexity of K.

In addition to the above properties, we shall be requiring another two important concepts, closedness of a cone and non empty interior of a cone, in the discussion. For

instance, $K = \{(x_1, x_2) \in R^2 : x_2 > |x_1|\} \cup \{0\}$ is a pointed convex cone with non empty interior but it is not a closed cone, however, $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 = x_2 \geq 0\}$ is a pointed closed convex cone with empty interior.

Summarizing the above discussion, what we find is that the set $\mathbf{R}^m_+$ in LPP is not significant but what is more important is the properties that this set possesses. A non empty pointed convex cone also possesses all the desired properties and thus can be a meaningful replacement of $\mathbf{R}^m_+$ in LPP (14.2). Thus, a general optimization problem involving cone is modeled as

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && Ax - b \geq_K 0. \end{aligned} \tag{14.4}$$

Problem of type (14.4) is called *conic programming problem* (CPP). From now onwards, we assume that the cone K in (14.4) is a closed pointed convex cone with non empty interior. LPP falls as a particular case of CPP with $K = \mathbf{R}^m_+$. The other two most interesting and widely studied cones are

m-dimensional Lorentz cone:

$$\begin{aligned} L^m &= \left\{(x_1, \ldots, x_{m-1}, x_m) \in \mathbf{R}^m : x_m \geq \sqrt{x_1^2 + \ldots + x_{m-1}^2}\right\} \\ &= \{(x, x_m) \in \mathbf{R}^m : x = (x_1, \ldots, x_{m-1}),\ x_m \geq \|x\|_2\}. \end{aligned}$$

This is also known as ice-cream cone or a second order cone. For $m = 1$, $L^m = R_+$, and for $m = 2$, $L^m = \{(x_1, x_2) \in R^2 : x_2 \geq |x_1|\}$, both are polyhedral cones. For $m \geq 3$, L^m is no longer a polyhedral cone.

Cone of positive semi-definite matrices:

$$S^m_+ = \{A \in \mathbf{R}^{m \times m} : A = A^T,\ A \succcurlyeq 0\},$$

where '$\succcurlyeq$' notation is used to represent the partial order of positive semi-definiteness. Thus, when we write $A \succcurlyeq 0$ we mean A is a positive semi-definite matrix.

One can easily verify that both the Lorentz cone and the cone of positive semi-definite matrices are non empty closed pointed convex cones with non empty interiors. In fact,

$$int(L^m) = \{(x, x_m) \in \mathbf{R}^m : x_m > \|x\|_2\},$$
$$int(S^m_+) = \{A \in \mathbf{R}^{m \times m} : A = A^T,\ A \succ 0\}.$$

Here, '$\succ$' stands for positive definiteness of a matrix.

14.4 Formulation of Semi-definite Programming Problem

We present certain basic concepts related to the underline set of matrices and the vector space structure imposed on it.

Let S^m be the set of $m \times m$ symmetric matrices. Under matrix addition and multiplication of a matrix by a real scalar, S^m, forms a finite dimensional real vector space with dimension ${}^{m+1}C_2$. As in $\mathbf{R}^m$, the inner product of two vectors is defined as their dot product, i.e. $\langle a, b\rangle = a^T b$, here, the inner product of two matrices in S^m is defined using *trace function*. So, for $U, V \in S^m$, we define

$$\langle U, V\rangle = \mathrm{Tr}(U^T V) = \mathrm{Tr}(UV),$$

where 'Tr$(\cdot)$' denotes the trace function and it is defined as the sum of diagonal elements of a matrix, i.e.

$$\mathrm{Tr}(U) = \mathrm{Tr}([u_{ij}]_{i,j=1}^{m}) = \sum_{i=1}^{m} u_{ii}.$$

This inner product is called *Frobenius inner product* on S^m.

We shall be using the upper script letter, like $\mathcal{A}$, to denote a linear mapping from $\mathbf{R}^n$ to S^m and Latin letter A to denote a matrix in S^m.

We are familiar with the linear mappings in $\mathbf{R}^n$. Generally, a linear mapping $\mathcal{T} : \mathbf{R}^n \to \mathbf{R}^m$ is described by

$$\mathcal{T}(x) = a_1 x_1 + \cdots + a_n x_n,$$

where $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$, and $a_i \in \mathbf{R}^m$ $(i = 1, \ldots, n)$.

For example, $\mathcal{T} : \mathbf{R}^2 \to \mathbf{R}^3$ defined as $\mathcal{T}(x_1, x_2) = (x_1 + x_2, x_1 - x_2, x_2)$, is equivalent to $\mathcal{T}(x_1, x_2) = a_1 x_1 + a_2 x_2$, with $a_1 = (1, 1, 0) \in \mathbf{R}^3$ and $a_2 = (1, -1, 1) \in \mathbf{R}^3$.

On parallel lines, we can specify a linear mapping $\mathcal{A} : \mathbf{R}^n \to S^m$ as

$$\mathcal{A}(x) = A_1 x_1 + \cdots + A_n x_n,$$

with $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$ and $A_i \in S^m$ $(i = 1, \ldots, n)$. For example, let $\mathcal{A} : \mathbf{R}^3 \to S^2$ be defined as

$$\begin{aligned}
\mathcal{A}(x) &= \begin{pmatrix} x_1 + x_2 & x_1 - x_2 \\ x_1 - x_3 & x_2 + x_3 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} x_1 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x_2 + \begin{pmatrix} 0 & -1 \\ -1 & 1 \end{pmatrix} x_3 \\
&= A_1 x_1 + A_2 x_2 + A_3 x_3.
\end{aligned}$$

Or if $\mathcal{A} : \mathbf{R}^2 \to S^3$ is

$$\begin{aligned}\mathcal{A}(x) &= \begin{pmatrix} -x_2 & x_1 & x_1+x_2 \\ x_1 & 0 & x_2 \\ x_1+x_2 & x_2 & x_1-x_2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} x_1 + \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \end{pmatrix} x_2 \\ &= A_1x_1 + A_2x_2.\end{aligned}$$

With this background we are ready to describe the optimization problem over the cone of positive semi-definite matrices. A general model of such conic optimization problem is given by

$$\begin{aligned} &\text{Min} && c^Tx \\ &\text{subject to} \end{aligned}$$

$$\mathcal{A}x - B \succcurlyeq_{S_+^m} 0. \tag{14.5}$$

Here $x \in \mathbf{R}^n$, $c \in \mathbf{R}^n$, $B \in S^m$, $\mathcal{A} : \mathbf{R}^n \to S^m$ is a linear mapping, and S_+^m is a cone of $m \times m$ symmetric positive semi-definite matrices. Moreover, when we say $U \succcurlyeq_{S_+^m} V$ we mean $U - V \in S_+^m$. In view of the above discussion on the linear mapping $\mathcal{A} : \mathbf{R}^n \to S^m$, problem (14.5) can be expressed as follows

$$\begin{aligned} &\text{Min} && c^Tx \\ &\text{subject to} \end{aligned}$$

$$A_1x_1 + \cdots + A_nx_n - B \succcurlyeq_{S_+^m} 0. \tag{14.6}$$

If we write, $\mathcal{F}(x) = A_1x_1 + \cdots + A_nx_n - B$, the problem is

$$\begin{aligned} &\text{Min} && c^Tx \\ &\text{subject to} \end{aligned}$$

$$\mathcal{F}(x) \succcurlyeq_{S_+^m} 0. \tag{14.7}$$

The inequality $\mathcal{F}(x) \succcurlyeq_{S_+^m} 0$ is called *linear matrix inequality* (LMI) and the conic optimization problem (14.7) is called *semi-definite programming problem* (SDPP).

Remark 14.4.1 *(i) As a convex combination of positive semi-definite matrices is positive semi-definite, consequently,*

$$\mathcal{F}(x) \succcurlyeq_{S_+^m} 0,\ \mathcal{F}(y) \succcurlyeq_{S_+^m} 0,\ \lambda \in [0,1] \Rightarrow \mathcal{F}(\lambda x + (1-\lambda)y) \succcurlyeq_{S_+^m} 0.$$

Thus SDPP (14.7) is a convex optimization problem.
(ii) If SDPP (14.7) is feasible then its optimal solution $x^\star$ is on the boundary of the feasible set. In other words, $\mathcal{F}(x^\star)$ is a singular matrix.

Remark 14.4.2 *While defining SDPP (14.7) only one LMI is imposed as a constraint. In fact we should not worry about more than one LMI in the constraint. The reason for this stems from the fact that the system of finitely many LMIs can easily be converted into a single LMI.*

For this, consider k number of LMIs as follows

$$\mathcal{A}_i(x) - B_i \succcurlyeq_{S_+^{m_i}} 0,$$

$\mathcal{A}_i : \mathbf{R}^n \to S^{m_i}$, $B \in S^{m_i}$ $(i = 1, \ldots, k)$.
Let $m_0 = m_1 + \cdots + m_k$. Define a linear mapping $\mathcal{A}_0 : \mathbf{R}^n \to S^{m_0}$ and a matrix $B_0 \in S^{m_0}$ as

$$\mathcal{A}_0(x) = \begin{pmatrix} \mathcal{A}_1(x) & 0_{m_1 \times m_2} & \cdots & 0_{m_1 \times m_k} \\ 0_{m_2 \times m_1} & \mathcal{A}_2(x) & \cdots & 0_{m_2 \times m_k} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{m_k \times m_1} & 0_{m_k \times m_2} & \cdots & \mathcal{A}_k(x) \end{pmatrix} = Diag(\mathcal{A}_1(x), \ldots, \mathcal{A}_k(x)),$$

$$B_0 = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_k \end{pmatrix} = Diag(B_1, \ldots, B_k).$$

Then the system of k LMIs, $\mathcal{A}_i(x) - B_i \succcurlyeq_{S_+^{m_i}} 0$ $(i = 1, \ldots, k)$, is equivalent to a single LMI, $\mathcal{A}_0(x) - B_0 \succcurlyeq_{S_+^{m_0}} 0$. To illustrate this result, consider the following system of LMIs

$$\begin{pmatrix} -x_1 + x_2 & x_1 - 3x_2 - 1 \\ x_1 - 3x_2 - 1 & 2x_2 \end{pmatrix} \succcurlyeq_{S_+^2} 0,$$

$$\begin{pmatrix} x_1 - x_2 - 1 & 2x_1 + x_2 + 1 \\ 2x_1 + x_2 + 1 & x_1 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

The above two LMIs can be combined into a single LMI given by

$$\begin{pmatrix} -x_1 + x_2 & x_1 - 3x_2 - 1 & 0 & 0 \\ x_1 - 3x_2 - 1 & 2x_2 & 0 & 0 \\ 0 & 0 & x_1 - x_2 - 1 & 2x_1 + x_2 + 1 \\ 0 & 0 & 2x_1 + x_2 + 1 & x_1 \end{pmatrix} \succcurlyeq_{S_+^4} 0.$$

This observation leads us to study SDPP (14.7) with one LMI only.
The next result is very important and is often used as the principal tool to reformulate the nonlinear programming problem as an appropriate SDPP.

Lemma 14.4.1 *Let* $A = \begin{pmatrix} M & C^T \\ C & D \end{pmatrix}$ *be a symmetric* $(l+k)\times(l+k)$ *matrix with* $k\times k$ *block* M, $l\times k$ *block* C *and* $l\times l$ *block* D. *Suppose* M *is a positive definite matrix. Then* A *is a positive semi-definite (respectively, positive definite) matrix if and only if* $D - CM^{-1}C^T$ *is a positive semi-definite (respectively, positive definite) matrix.*

Proof. The positive semi-definiteness of A is equivalent to saying

$$(x^T \; y^T)\begin{pmatrix} M & C^T \\ C & D \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \geq 0, \quad \forall\, x \in \mathbf{R}^k,\ y \in \mathbf{R}^l,$$

i.e.

$$x^T M x + 2x^T C^T y + y^T D y \geq 0, \quad \forall\, x \in \mathbf{R}^k,\ y \in \mathbf{R}^l.$$

The above inequality yields

$$\inf_{x \in \mathbf{R}^k} (x^T M x + 2x^T C^T y + y^T D y) \geq 0, \quad \forall\, y \in \mathbf{R}^l. \tag{14.8}$$

Now, for every fixed $y \in \mathbf{R}^l$, the infimum of the above problem is attained at, $x^\star = -M^{-1}C^T y$, and the minimum value is $y^T(D - CM^{-1}C^T)y$. It is important to remark here that the positive definiteness of M is used to infer that $x^\star$ is an optimal solution of problem (14.8).

Thus, the positive semi-definiteness (respectively, positive definiteness) of A is equivalent to $y^T(D - CM^{-1}C^T)y \geq 0,\ \forall\, y \in \mathbf{R}^l$ (respectively, $y^T(D - CM^{-1}C^T)y > 0,\ \forall\, y \in \mathbf{R}^l,\ y \neq 0$). In other words, it is same as saying that

$$A \succcurlyeq_{S_+^{l+k}} 0 \text{ (respectively, } \succ_{S_+^{l+k}} 0) \Leftrightarrow D - CM^{-1}C^T \succcurlyeq_{S_+^l} 0 \text{ (respectively, } \succ_{S_+^l} 0). \quad \square$$

Remark 14.4.3 *The matrix* $D - CM^{-1}C^T$ *is called Schur complement of* M *in* A, *named after its discoverer, Issai Schur.*

14.5 Applications

At this juncture we would like to see what kind of problems can be casted in the form of SDPP. Consider the LPP

$$\begin{aligned} \text{Min} \quad & c^T x \\ \text{subject to} \quad & \\ & Ax \geq b, \end{aligned}$$

$A \in \mathbf{R}^{m\times n},\ x \in \mathbf{R}^n,\ b \in \mathbf{R}^m.$

Define

$$A_i = Diag(a_i) = \begin{pmatrix} a_{1i} & 0 & \cdots & 0 \\ 0 & a_{2i} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{mi} \end{pmatrix} \quad (i = 1, \ldots, n),$$

$$B = Diag(b) = \begin{pmatrix} b_1 & 0 & \cdots & 0 \\ 0 & b_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_m \end{pmatrix},$$

and let $\mathcal{F}(x) = \sum_{i=1}^{n} A_i x_i - B$. Then LPP can be expressed as a SDPP

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && \mathcal{F}(x) \succcurlyeq_{S_+^m} 0. \end{aligned}$$

For example, consider the LPP

$$\begin{aligned} &\text{Min} && 2x_1 + 3x_2 \\ &\text{subject to} \\ & && x_1 + x_2 \leq 6 \\ & && -x_1 + 2x_2 \leq 8 \\ & && x_1, x_2 \geq 0. \end{aligned}$$

Taking $A_1 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} -6 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, the given LPP can be written as

$$\begin{aligned} &\text{Min} && 2x_1 + 3x_2 \\ &\text{subject to} \\ & && \mathcal{F}(x) = \begin{pmatrix} -x_1 - x_2 + 6 & 0 & 0 & 0 \\ 0 & x_1 - 2x_2 + 8 & 0 & 0 \\ 0 & 0 & x_1 & 0 \\ 0 & 0 & 0 & x_2 \end{pmatrix} \succcurlyeq_{S_+^4} 0. \end{aligned}$$

Another important class of nonlinear programming problems that arise frequently in combinatorial optimization problems and can be formulated as SDPP is the class of quadratically constrained quadratic programming problems

$$\begin{aligned} &\text{Min} && x^T Q_0 x + c_0^T x + d_0 \\ &\text{subject to} \\ & && x^T Q_1 x + c_1^T x + d_1 \leq 0. \end{aligned} \tag{14.9}$$

Here, $Q_i \in S^n$ $(i = 0, 1)$ are positive definite matrices, $c_i \in \mathbf{R}^n$, $d_i \in \mathbf{R}$ $(i = 0, 1)$, $x \in \mathbf{R}^n$. By introducing an additional variable the problem can be converted into an optimization problem with linear objective function

$$\begin{aligned} &\text{Min} && \xi \\ &\text{subject to} \\ & && x^T Q_0 x + c_0^T x + d_0 \leq \xi \\ & && x^T Q_1 x + c_1^T x + d_1 \leq 0. \end{aligned} \tag{14.10}$$

Here we take a small pause to recall from eigen value characterization in matrix theory that if U is a symmetric positive definite matrix then there exist an orthogonal matrix M and a diagonal matrix Λ with positive entries λ_j such that $U = M\Lambda M^T$. Define $\Lambda^{1/2} = Diag(\lambda_j^{1/2})$, and $U^{1/2} = M\Lambda^{1/2}M^T$. Then $U^{1/2}$ is a positive definite matrix and $U^{1/2}U^{1/2} = U$.

From the above observation, we infer that there exist positive definite matrices P_i $(i = 0, 1)$, such that $Q_i = P_i{}^T P_i$ $(i = 0, 1)$. So, the constraints in (14.10) become

$$x^T P_i{}^T I P_i x \leq \xi_i - c_i^T x - d_i \quad (i = 0, 1), \quad \xi_0 = \xi, \ \xi_1 = 0.$$

Applying Lemma 14.4.1, the two inequalities can be reframed as

$$\begin{pmatrix} I_{n\times n} & P_0 x & 0_{n\times n} & 0_{n\times 1} \\ x^T P_0{}^T & \xi_0 - c_0^T x - d_0 & 0_{1\times n} & 0_{1\times 1} \\ 0_{n\times n} & 0_{n\times 1} & I_{n\times n} & P_1 x \\ 0_{1\times n} & 0_{1\times 1} & x^T P_1{}^T & -c_1^T x - d_1 \end{pmatrix} \succcurlyeq_{S_+^{2n+2}} 0.$$

Thus, problem (14.9) is equivalent to a SDPP.

Remark 14.5.1 *The inequality of the form* $x^T P^T P x + c^T x + d \leq 0$ *along with Lemma 14.4.1 yields* $\begin{pmatrix} I & Px \\ x^T P^T & -(c^T x + d) \end{pmatrix} \succcurlyeq_{S_+^{n+1}} 0$, *irrespective of the nature of the matrix* P.

Recall the example from Section 14.2

$$\begin{aligned} &\text{Min} && x_2 - x_1 \\ &\text{subject to} \\ & && x_1^2 + x_2^2 \leq 2 \\ & && x_2^2 - x_1 \leq 0. \end{aligned}$$

The two constraints are

$$x_1^2+x_2^2-2 = \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 2 \leq 0$$

$$x_2^2 - x_1 = \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} -1 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq 0.$$

In view of the Remark 14.5.1, the given nonlinear programming problem is equivalent to the following SDPP

$$\begin{array}{ll} \text{Min} & x_2 - x_1 \\ \text{subject to} & \end{array}$$

$$\begin{pmatrix} 1 & 0 & x_1 & 0 & 0 & 0 \\ 0 & 1 & x_2 & 0 & 0 & 0 \\ x_1 & x_2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_2 \\ 0 & 0 & 0 & 0 & x_2 & x_1 \end{pmatrix} \succcurlyeq_{S_+^6} 0.$$

Another interesting problem that can be transformed as a SDPP is

$$\begin{array}{ll} \text{Min} & \dfrac{(c^T x)^2}{d^T x} \\ \text{subject to} & \\ & Ax \geq b, \end{array}$$

with an assumption that $d^T x > 0$ whenever $Ax \geq b$.
The above problem is equivalent to

$$\begin{array}{ll} \text{Min} & \xi \\ \text{subject to} & \end{array}$$

$$\begin{array}{rl} Ax \geq & b \\ (c^T x)^2 \leq & \xi d^T x. \end{array}$$

Using the Schur complement (Lemma 14.4.1), the constraints can be reformulated as LMI in the variables x and ξ as

$$\begin{pmatrix} Diag(Ax-b) & 0 & 0 \\ 0 & \xi & c^T x \\ 0 & c^T x & d^T x \end{pmatrix} \succcurlyeq_{S_+^{m+2}} 0.$$

Here $Diag(Ax-b)$ means an $m \times m$ diagonal matrix in which the i-th diagonal entry is determined by the i-th linear constraint.

Beside the listed classes of problems, semi-definite programming (SDP) can handle more complicated nonlinear programming problems. We now present few examples that

are no way exhaustive but provide some insight into the ability of SDP in handling various situations.

To begin with we present the most common eigen value problem that can be beautifully formulated as SDPP. The problem is to find minimum of the maximum eigen value of a matrix. Mathematically, the problem can be posed as follows.

Suppose $\mathcal{A} : \mathbf{R}^n \to S^m$ is a linear mapping. Find $z^\star \in \mathbf{R}^n$ such that $z^\star$ solves the optimization problem

$$\text{Min} \ \ \lambda_{\max}(\mathcal{A}(z)), \tag{14.11}$$

where $\lambda_{\max}(\mathcal{A}(z))$ denote the maximum eigen value of a symmetric matrix $\mathcal{A}(z)$. Note that

$$\begin{array}{rrcl} & \lambda_{\max}(\mathcal{A}(z)) & \leq & \eta \\ \Leftrightarrow & \lambda_{\min}(\eta I_{m\times m} - \mathcal{A}(z)) & \geq & 0 \\ \Leftrightarrow & \eta I_{m\times m} - \mathcal{A}(z) & \succcurlyeq_{S^m_+} & 0. \end{array}$$

Thus problem (14.11) can equivalently be posed as

$$\begin{array}{ll} \text{Min} & \eta \\ \text{subject to} & \\ & \eta I_{m\times m} - \mathcal{A}(z) \succcurlyeq_{S^m_+} \ 0. \end{array}$$

Another related problem is to

$$\text{Min} \ \ \|\mathcal{A}(x)\|_2,$$

$\mathcal{A} : \mathbf{R}^n \to \mathbf{R}^{p\times q}$ is a linear mapping and $x \in \mathbf{R}^n$. Observe that here $\mathcal{A}(x)$ need not be a symmetric matrix. This problem can be casted as SDPP

$$\begin{array}{ll} \text{Min} & \eta \\ \text{subject to} & \\ & \begin{pmatrix} \eta I_{p\times p} & \mathcal{A}(x) \\ \mathcal{A}(x)^T & \eta I_{q\times q} \end{pmatrix} \succcurlyeq_{S^{p+q}_+} \ 0. \end{array}$$

One has to realize that though the original problems are not actually in the form of SDPPs but by intelligently using the Schur complement they can be modeled as SDPPs. Thus, recognizing the Schur complement in a nonlinear expression is the key step in reformulating a nonlinear optimization problem as SDPP.

To appreciate the power of SDP we mention few more areas where the problems have been skillfully modeled as SDPPs. For instance, structural designs in mechanical constructions, control and system problems dealing with differential equations and their solutions stability analysis, geometrical problems of volumes and ellipsoid approximations, trace factor analysis problems arising in statistics, eigen value problems in linear algebra, nonconvex optimization problems involving polynomial objective and polynomial constraints, problems of pattern classification and support vector machines. To

know about these problems and their formulation as SDPPs, we recommend the text by Ben-Tal and Nemrovski [16].

In particular, we would like to sketch the role of SDP in machine learning. The reason behind this is that later a complete chapter (Chapter 16) is devoted to 'machine learning'. It is then and there that we shall be carrying out the detail analysis of pattern classification problems. The principal problem of pattern classification is to classify two sets of patterns by finding an appropriate separating hyperplane. In many cases the requisite hyperplane is nonlinear. The problem is transformed into an optimization problem involving a quadratic objective function. The 'kernal function' contributes to the quadratic term. Subsequently, the primary optimization problem implicitly involves another problem of choosing the right 'kernel function'. The latter is a hard problem to crack and it has led to the emergence of new area of research called *kernal optimization*. When the problem of kernal optimization is translated into an optimization model one came across SDPP very naturally. Consequently the problem of kernal optimization is addressed to via SDP. This area is still very young and a lot is needed to be done before drawing any conclusion. But our idea is to highlight that, besides the traditional fields, SDP is in the center of many emerging areas. Of course one can easily extract more information on SDP and its applications in various areas through the vast literary material available on this topic.

14.6 Formulation of the Dual Problem

It is time to address the theoretical issues related to SDP. One of the most elegant theoretical result in LP is the LP duality. Since SDP can be regarded as an extension of LP therefore it is natural to raise an issue that if such duality theory can be developed for SDP. We shall witness in this section that the duality theory for SDP is closely parallel to the duality theory for LPP but there are some important differences too. Before proceeding further we must get familiar with the concept of *dual cone*.

Definition 14.6.1 (Dual Cone). *Suppose K is a non empty subset of $\mathbf{R}^p$. Then the set $K^\star$ defined as*

$$K^\star = \{v \in \mathbf{R}^p : \langle v, k\rangle \geq 0,\ \forall\, k \in K\}$$

is called the dual cone of K.

Remark 14.6.1 *It is important to note that, in definition 14.6.1, the set K is not required to be a cone but $K^\star$ is always a non empty closed convex cone.*

For example, if

(*i*) $K = \mathbf{R}^2_+$ then $K^\star = \mathbf{R}^2_+$;
(*ii*) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1^2 + x_2^2 \leq 1\}$ then $K^\star = \{0\}$;
(*iii*) $K = \{(1,0), (0,1)\}$ then $K^\star = \mathbf{R}^2_+$;
(*iv*) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq 0\}$ then $K^\star = \{(v_1, v_2) \in \mathbf{R}^2 : v_1 = 0, v_2 \geq 0\}$;
(*v*) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 \geq x_2 \geq 0\}$ then $K^\star = \{(v_1, v_2) \in \mathbf{R}^2 : v_1 \geq 0, v_1 + v_2 \geq 0\}$.

In the definition to follow we assume that the set K is a cone.

Definition 14.6.2 (Self Dual Cone). *A cone K is said to be a self dual cone if $K = K^\star$.*

For example, $\mathbf{R}^m_+$ is a self dual cone and so is the cone $\{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq |x_1|\}$ but $\{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq 0\}$ is not a self dual cone.

Lemma 14.6.1 *The cone of symmetric positive semi-definite matrices is a self dual cone, i.e. $S^m_+ = (S^m_+)^\star$.*

Proof. Recall that the inner product in the vector space S^m is defined as $\langle U, V \rangle = Tr(UV)$. So,

$$(S^m_+)^\star = \{U \in S^m : Tr(UV) \geq 0, \ \forall \ V \in S^m_+\}.$$

We first prove that $(S^m_+)^\star \subseteq S^m_+$. Let $U, V \in S^m_+$. Then we can express $U = M\Lambda M^T$, where M is an orthogonal matrix and Λ is a diagonal matrix with nonnegative diagonal entries. Then

$$\langle U, V \rangle = Tr(UV) = Tr(M\Lambda M^T V) = Tr((M^T V M)\Lambda).$$

Note that the last equality follows on account of $Tr(AB) = Tr(BA)$. Now, $y^T M^T V M y = (My)^T V (My) \geq 0, \ \forall \ y \in \mathbf{R}^m$, so, $M^T V M$ is a positive semidefinite matrix. Thus its diagonal entries are nonnegative and Λ is a diagonal matrix with non negative entries. Consequently, $Tr(UV) \geq 0$. Thereby yielding the desired containment.

Conversely, let $U \in (S^m_+)^\star$. Then

$$Tr(UV) \geq 0, \quad \forall \ V \in S^m_+.$$

Let $w \in \mathbf{R}^m$. Choose $V = ww^T \in S^m$. Then, $x^T V x = (w^T x)^2 \geq 0, \ \forall \ x \in \mathbf{R}^m$. Thus $V \in S^m_+$. With this choice of V,

$$Tr(UV) = Tr(Uww^T) = Tr(w^T U w) = w^T U w \geq 0 \Rightarrow U \in S^m_+.$$

$\Rightarrow (S^m_+)^\star \subseteq S^m_+$. Completing the requisite equality of the two sets. □

Similar to the case $(\mathbf{R}^m_+)^\star = \mathbf{R}^m_+$ in LPP, we have, $(S^m_+)^\star = S^m_+$. It is thus expected that the duality theory for SDP will follow on the lines of LP duality. It is shown to be partially correct in the discussion to follow.

We briefly turn around and take a look at what we studied in LP duality. If the primal LPP is of the form

$$\begin{array}{ll} \text{Min} & c^T x \\ \text{subject to} & \\ & Ax \geq b, \end{array} \tag{14.12}$$

then the associated dual is

$$\begin{array}{ll} \text{Max} & b^T \lambda \\ \text{subject to} & \\ & A^T \lambda = c \\ & \lambda \geq 0. \end{array} \tag{14.13}$$

Taking inspiration from the primal-dual models, (14.12)-(14.13), we construct the dual of the following SDPP

$$\begin{array}{ll} \text{Min} & c^T x \\ \text{subject to} & \\ & \mathcal{A}(x) \succcurlyeq_{S^m_+} B. \end{array} \tag{14.14}$$

Since the constraint is in the form of LMI it is immediate that the dual variable (or the Lagrange multiplier) shall be a matrix, say Λ. Consequently the dot product $b^T\lambda$ between the two vectors b and λ in the objective function of (14.13) should be replaced by the inner product between two matrices, $\langle B, \Lambda \rangle = Tr(B\Lambda)$. Further, A^T is the conjugate of the matrix A in (14.13), so in context of SDPP it is to be replaced by the conjugate of the linear mapping $\mathcal{A}$, denoted by $\mathcal{A}^d$. Remember that the constraint inequality $Ax \geq b$ in (14.12) is same as $Ax - b \in \mathbf{R}^m_+$, and thus $\lambda \in (\mathbf{R}^m_+)^\star = \mathbf{R}^m_+$. Taking into account that the cone S^m_+ is a self dual cone, we have, $\Lambda \in S^m_+$. Putting all these facts together, the dual to SDPP (14.14) is another SDPP given by

$$\begin{array}{ll} \text{Max} & Tr(B\Lambda) \\ \text{subject to} & \\ & \mathcal{A}^d(\Lambda) = c \\ & \Lambda \succcurlyeq_{S^m_+} 0. \end{array} \tag{14.15}$$

We next analyze the equation $\mathcal{A}^d(\Lambda) = c$. For a linear map $\mathcal{A} : \mathbf{R}^n \to S^m$, the conjugate map $\mathcal{A}^d : S^m \to \mathbf{R}^n$ is described by

$$\langle \mathcal{A}(x), \Lambda \rangle = \langle x, \mathcal{A}^d(\Lambda) \rangle, \quad x \in \mathbf{R}^n, \ \Lambda \in S^m. \tag{14.16}$$

An important point to observe is that the inner product on the left side of (14.16) is in S^m while the inner product on the right side is in R^n. Now,

$$\begin{array}{rcl} \langle \mathcal{A}(x), \Lambda \rangle & = & Tr(\mathcal{A}(x)\Lambda) \\ & = & Tr((A_1 x_1 + \cdots + A_n x_n)\Lambda) \\ & = & x_1 Tr(A_1\Lambda) + \cdots + x_n Tr(A_n\Lambda). \end{array} \tag{14.17}$$

From (14.16) and (14.17) it follows that

$$(\mathcal{A}^d(\Lambda))^T x = (Tr(A_1\Lambda), \ldots, Tr(A_n\Lambda))x,$$

thereby yielding that

$$\mathcal{A}^d(\Lambda) = \begin{pmatrix} Tr(A_1\Lambda) \\ \vdots \\ Tr(A_n\Lambda) \end{pmatrix}.$$

Finally the dual to the SDPP (14.14) is given by

Max $\quad Tr(B\Lambda)$

subject to

$$\begin{aligned} Tr(A_j\Lambda) &= c_j \qquad (j = 1, \ldots, n) \\ \Lambda &\succcurlyeq_{S_+^m} 0. \end{aligned} \tag{14.18}$$

Example 14.6.1 *Construct the dual of the following SDPP*

Min $\quad 2x_2$

subject to

$$\begin{pmatrix} x_1 - 1 & x_2 \\ x_2 & x_1 - 2x_2 + 1 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

Solution The constraint of the SDPP can be expressed as LMI

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x_1 + \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} x_2 - \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

Let $\Lambda = \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{12} & \lambda_{22} \end{pmatrix}$. After simple calculations we can see that

$$Tr(B\Lambda) = \lambda_{11} - \lambda_{22},\ Tr(A_1\Lambda) = \lambda_{11} + \lambda_{22},\ Tr(A_2\Lambda) = 2\lambda_{12} - 2\lambda_{22},$$

hence, the dual of the primal SDPP is given by

Max $\quad \lambda_{11} - \lambda_{22}$

subject to

$$\begin{aligned} \lambda_{11} + \lambda_{22} &= 0 \\ \lambda_{12} - \lambda_{22} &= 1 \\ \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{12} & \lambda_{22} \end{pmatrix} &\succcurlyeq_{S_+^2} 0. \end{aligned}$$

Like in LPP, we have the following table depicting the primal-constraint dual-variable relationships in SDPP

Primal: Min		Dual: Max
Variable type		Constraint type
$x \geq_{\mathbf{R}^n_+} 0$	$\leftrightarrow$	$\leq_{\mathbf{R}^n_+}$
$\Lambda \succcurlyeq_{S^m_+} 0$	$\leftrightarrow$	$\preccurlyeq_{S^m_+}$
unrestricted	$\leftrightarrow$	$=$
Constraint type		Variable type
$\succcurlyeq_{S^m_+}$	$\leftrightarrow$	$\Lambda \succcurlyeq_{S^m_+} 0$
$\geq_{\mathbf{R}^n_+}$	$\leftrightarrow$	$x \geq_{\mathbf{R}^n_+} 0$
$=$	$\leftrightarrow$	unrestricted

The above table can be used to construct the dual programs of various SDP models.

14.7 Duality Theorems

In this section we describe the duality relations between the following pair of primal SDPP

$$\begin{aligned} &\text{Min} && c^T x \\ &\text{subject to} \\ & && \mathcal{A}(x) \succcurlyeq_{S^m_+} B \end{aligned} \tag{14.19}$$

and its dual SDPP

$$\begin{aligned} &\text{Max} && Tr(B\Lambda) \\ &\text{subject to} \\ & && \mathcal{A}^d_j(\Lambda) = c_j \quad (j = 1, \ldots, n) \\ & && \Lambda \succcurlyeq_{S^m_+} 0. \end{aligned} \tag{14.20}$$

Theorem 14.7.1 *SDPP is a symmetric problem, i.e. the dual of SDPP (14.20) is the primal SDPP (14.19).*

Proof. Writing the dual SDPP (14.20) in the following form

$$\begin{aligned} &\text{- Min} && -Tr(B\Lambda) \\ &\text{subject to} \\ & && \mathcal{A}^d(\Lambda) \geq c \\ & && -\mathcal{A}^d(\Lambda) \geq -c \\ & && \Lambda \succcurlyeq_{S^m_+} 0. \end{aligned} \tag{14.21}$$

We can write the dual of (14.21) as follows

$$\begin{aligned} -\text{Max} \quad & c^T y_1 - c^T y_2 \\ \text{subject to} & \end{aligned}$$

$$\begin{aligned} \mathcal{A}(y_1) - \mathcal{A}(y_2) + Tr(\Lambda W) &= -B \\ y_1,\, y_2 &\geq 0 \\ W &\succcurlyeq_{S_+^m} 0. \end{aligned} \tag{14.22}$$

Taking $y_2 - y_1 = y$, using linearity of functions $\mathcal{A}(\cdot)$ and $Tr(\cdot)$ and the fact that $Tr(\Lambda W) \geq 0$, (14.22) can be rewritten as

$$\begin{aligned} \text{Min} \quad & c^T y \\ \text{subject to} & \end{aligned}$$

$$\mathcal{A}(y) \succcurlyeq_{S_+^m} B.$$

The above problem is same as the primal SDPP (14.19). □

Theorem 14.7.2 (Weak Duality Theorem). *Let x be feasible for primal SDPP (14.19) and Λ be feasible for the dual SDPP (14.20). Then $c^T x \geq Tr(B\Lambda)$.*

Proof. Since $\mathcal{A}(x) - B \succcurlyeq_{S_+^m} 0$ and $\Lambda \succcurlyeq_{S_+^m} 0$, we have,

$$\begin{aligned} &\langle \mathcal{A}(x) - B, \quad \Lambda \rangle \geq 0, \\ \Rightarrow\ & Tr(B\Lambda) \geq \langle \mathcal{A}(x),\, \Lambda \rangle \\ & \geq \langle x,\, \mathcal{A}^d(\Lambda) \rangle \\ & \geq \langle x,\, c \rangle. \end{aligned}$$

□

Definition 14.7.1 (Duality Gap). *Let x be a feasible solution of the primal SDPP (14.19) and Λ be a feasible solution of the dual SDPP (14.20). The difference between the objective values, $c^T x - Tr(B\Lambda)$, is called the duality gap between (14.19) and (14.20).*

So far the duality results for SDPP follow nicely on the similar pattern as we have for LPP. However, the major difference will arise at this stage when we talk about the strong duality ensuring zero duality gap between (14.19) and (14.20). Unlike LPP, strong duality does not follow very easily for SDPP. We first provide few examples that will reveal the difficulties arising frequently in establishing the strong duality result for SDPP. A closer and careful look at these examples is necessary to identify the conditions under which the strong duality result can be worked out.

Example 14.7.1 *Write the dual of the following SDPP (primal)*

$$\begin{aligned} \textit{Min} \quad & x_1 \\ \textit{subject to} & \end{aligned}$$

$$X = \begin{pmatrix} 0 & x_1 & 0 \\ x_1 & x_2 & 0 \\ 0 & 0 & x_1 + 1 \end{pmatrix} \succcurlyeq_{S_+^3} 0.$$

Obtain the optimal solutions of the primal SDPP and its dual problem. Show that there exists a positive duality gap between the two problems.

Solution Since X is a positive semi-definite matrix, all principal sub-minors of X are non-negative, thereby yielding $x_1 = 0$. Thus the optimal value of this problem is 0 and the optimal solution is $\begin{pmatrix} 0 & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $x \in \mathbf{R}$.

The corresponding dual SDPP is given by

$$\begin{aligned} &\text{Max} && -\lambda_{33} \\ &\text{subject to} \end{aligned}$$

$$\begin{pmatrix} \lambda_{11} & \frac{1-\lambda_{33}}{2} & \lambda_{13} \\ \frac{1-\lambda_{33}}{2} & 0 & -\lambda_{23} \\ \lambda_{13} & \lambda_{23} & \lambda_{33} \end{pmatrix} \succcurlyeq_{S_+^3} 0.$$

From the constraint of the dual SDPP we infer, $-(1-\lambda_{33})^2/4 \geq 0$, implying $\lambda_{33} = 1$. The feasible set of the dual SDPP is

$$\left\{ \begin{pmatrix} 0 & 0 & \lambda_{13} \\ 0 & 0 & \lambda_{23} \\ \lambda_{13} & \lambda_{23} & 1 \end{pmatrix} : \lambda_{13}, \lambda_{23} \in \mathbf{R} \right\} \bigcup \left\{ \begin{pmatrix} \lambda_{11} & 0 & \lambda_{13} \\ 0 & 0 & 0 \\ \lambda_{13} & 0 & 1 \end{pmatrix} : \lambda_{11} > 0, \lambda_{13} \in \mathbf{R} \right\}.$$

Hence the optimal value of the dual objective is -1.

Observe that the duality gap is 1 and not 0 unlike in LPP where there is no duality gap at the optimal solutions.

In the problem to follow both the primal SDPP and its dual SDPP have equal optimal values but the primal SDPP is not solvable.

Example 14.7.2 *Consider the primal SDPP*

$$\begin{aligned} &\textit{Inf} && x_1 \\ &\textit{subject to} \end{aligned}$$

$$\begin{pmatrix} x_1 & 1 \\ 1 & x_2 \end{pmatrix} \succcurlyeq_{S_+^2} 0$$

Write the dual of the primal SDPP. Show that the optimal values of the primal SDPP and its dual are equal, but the primal SDPP do not possess an optimal solution.

Solution The dual SDPP is as follows

$$\begin{aligned} &\text{Sup} && -2\lambda_{12} \\ &\text{subject to} \end{aligned}$$

$$\begin{pmatrix} 1 & \lambda_{12} \\ \lambda_{12} & 0 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

The constraints of the primal-dual pair yield that $x_1 \geq 0$, $x_1 x_2 \geq 1$, and $-\lambda_{12}^2 \geq 0$. Thus the optimal values of both the primal problem and its dual problem are 0 implying zero duality gap. However, the lower bound of the primal problem is $1/x_2$ which goes to 0 as $x_2 \to \infty$. Consequently the primal SDPP is not solvable as its optimal value is not attainable.

The next problem is included to depict that even though the primal SDPP is infeasible the dual SDPP is solvable with finite optimal value.

Consider the primal SDPP

$$\begin{aligned} \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle &= 0 \\ \left\langle \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, X \right\rangle &= 2 \\ X &\succcurlyeq_{S_+^2} 0. \end{aligned}$$

The objective function can be taken as a zero function, i.e. Max $\left\langle \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle$. A matrix X satisfying the first two constraints of the problem is of the form $\begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix}$. But such a matrix can not be a positive semi-definite matrix. So the SDPP is infeasible.

The dual of this problem is the following SDPP

$$\begin{aligned} &\text{Min} && 2\lambda_2 \\ &\text{subject to} \\ & && \begin{pmatrix} \lambda_1 & \lambda_2 \\ \lambda_2 & 0 \end{pmatrix} \succcurlyeq_{S_+^2} 0. \end{aligned}$$

The set of optimal solutions of the dual SDPP is $\left\{ \begin{pmatrix} \lambda_1 & 0 \\ 0 & 0 \end{pmatrix} : \lambda_1 \geq 0 \right\}$, and the optimal value is 0.

The above examples clearly demonstrate that feasibility or boundedness of either of the problem does not imply solvability of the other problem and/or equality between their optimal values. One of the reason behind this phenomena is that S_+^m is not a polyhedral cone and thus word by word extension of LPP duality is not possible. It also inspires us to look for the sufficient conditions that ensure zero duality gap between the primal SDPP (14.19) and its dual SDPP (14.20). The search leads us to the following concept.

Definition 14.7.2 *(i) The primal SDPP (14.19) is said to be strictly feasible if there exists $x \in \mathbf{R}^n$ such that $\mathcal{A}(x) - B \succ_{S_+^m} 0$.*
(ii) The dual SDPP (14.20) is said to be strictly feasible if there exists $\Lambda \succ_{S_+^m} 0$ such that $\mathcal{A}^d(\Lambda) = c$.

For instance, in Example 14.7.1, the feasible solutions set of the primal SDPP is described by $\left\{\begin{pmatrix} 0 & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & 1 \end{pmatrix} : x \in \mathbf{R}\right\}$. The matrix of this type can not be a positive definite matrix. Thus, the primal SDPP is not strictly feasible. Similar arguments can be given to justify that the corresponding dual SDPP is also not strictly feasible. One can easily verify that in the other two examples presented above neither the primal SDPP nor its dual SDPP are strictly feasible. We now present an example in which both the primal SDPP and the dual SDPP are strictly feasible consequently yielding zero duality gap.

Example 14.7.3 *Consider the following SDPP (primal)*

$$\begin{aligned} &\textit{Min} && x_1 + x_2 \\ &\textit{subject to} \end{aligned}$$

$$\begin{pmatrix} x_1 & x_2 \\ x_2 & 1 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

Obtain the optimal solutions of the primal SDPP and its dual. Show that the optimal values of both problems coincide.

Solution The feasibility condition can be reframed as the system of nonlinear inequalities, $x_1 \geq 0,\ x_1 - x_2^2 \geq 0$, thereby providing the optimal solution of SDPP as $\begin{pmatrix} 1/4 & -1/2 \\ -1/2 & 1 \end{pmatrix}$ and the optimal value is $-1/4$.

The dual SDPP is given by

$$\begin{aligned} &\text{Max} && -\lambda_{22} \\ &\text{subject to} \end{aligned}$$

$$\begin{aligned} \lambda_{11} &= 1 \\ 2\lambda_{12} &= 1 \\ \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{12} & \lambda_{22} \end{pmatrix} &\succcurlyeq_{S_+^2} 0. \end{aligned}$$

The feasible set of the dual SDPP is $\left\{\begin{pmatrix} 1 & 1/2 \\ 1/2 & \lambda_{22} \end{pmatrix} : \lambda_{22} \geq \frac{1}{4}\right\}$. Clearly the optimal solution is $\begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/4 \end{pmatrix}$ and the optimal value is $-1/4$.

Here both the primal SDPP and the dual SDPP are strictly feasible and possess optimal solutions with equal optimal values.

Theorem 14.7.3 (Strong Duality Theorem). *If the primal SDPP (14.19) is bounded below and strictly feasible then the dual SDPP (14.20) is solvable and the optimal values of the two problems are equal.*

The proof of this theorem is skipped as it uses the 'separation theorem' of convex sets. So, the knowledge of the 'separation theorem' is very much required to understand the working of the proof of this theorem. This topic is not covered in the book. However, keen readers can refer to Ben-Tal and Nemrovski [16].

The following remark is a consequence of Theorem 14.7.1 and Theorem 14.7.3.

Remark 14.7.1 *If at least one of the SDPP (14.19) or (14.20) is bounded and strictly feasible then the two SDPPs are solvable and the optimal values of the two problems are equal.*

14.8 Summary and Additional Notes

- This Chapter introduces a new class of optimization problems, namely, semi-definite programming problems (SDPPs). This class of problems originated from the idea of defining partial order using the concept of cone. The related background is build in the initial two sections, Sections 14.2-14.3.
- In Section 14.4, we formulated a general model of SDPP via linear matrix inequality. It is shown in Section 14.5 that several classes of problems can be casted as SDPPs, thereby, strengthens the significance of the class of SDPPs.
- Sections 14.6-14.7 are devoted to describe the dual formulation and duality results for SDPP. It is noted that the strong duality result in SDPP does not follow naturally and requires additional conditions on the feasible sets of the primal-dual pair of problems.
- In recent years, much effort has been undertaken to obtain the strong duality result for SDPP. The most prominent among them are the contributions of Ramana et al. [130] and Yang [169]. Very recently Jeyakumar [84] presented new necessary and sufficient conditions for the strong duality in SDPP.
- Unlike LPP, where the interior point software development begins in the mid eighties, for SDPP it all started only in mid ninties. In the last decade it has grown manifold. The codes currently available can solve some thousand variables SDPP. Although this figure is not compatible with that of LPP interior point software which can handle hundred of thousands of variables yet we find no reason to be disappointed. With the subject witnessing tremendous theoretical growth and with the rapid advancement in the technology, we strongly trust to see significant progress in the coming years. Information on interior point softwares for SDPP can be obtained by following the link on the web page http://www-user.tu-chemnitz.de/~helmberg/semidef.html.
- Although numerous research articles are available on SDPP but for the beginners we recommend the excellent books by Ben-Tal and Nemrovski [16] and Wolkowicz et al. [168]. The research articles by Alizadeh [1], Todd [159], Vandenberghe and Boyd [160] are worth looking.

14.9 Exercises

14.1 *Let $c \in \mathbf{R}^n$ and $\alpha \in \mathbf{R}$, $\alpha \neq 0$. Define*

$$K = \{x \in \mathbf{R}^n : c^T x \geq 0\}, \quad K_1 = \{x \in \mathbf{R}^n : c^T x \geq \alpha\}.$$

Show that K is a closed convex cone but it is not pointed, while K_1 is not a cone.

14.2 *Discuss which of the following sets are closed pointed convex cones*

(i) $K = \{(x_1, x_2, x_3) \in \mathbf{R}^3 \mid x_1 \geq x_2 \geq x_3 \geq 0\}$.

(ii) $K = \{0\} \cup \{x \in \mathbf{R}^n \mid x_1 = x_2 = \cdots = x_k = 0, \text{ for some } k \ (k = 1, \ldots, n-1)\}$.

(iii) $K = \{(x, t) \in \mathbf{R}^n \times \mathbf{R}_+ \mid x^T x \leq t^2\}$.

(iv) $K = \{c \in \mathbf{R}^n : c_1 + c_2 x + c_3 x^2 + \cdots + c_n x^{n-1} \geq 0, \ \forall\, x \in [0, 1]\}$.

(v) $K = \{A \in S^{n \times n} : x^T A x \geq 0, \ \forall\, x \geq 0\}$.

14.3 *Find the dual cone of the following cones*

(i) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_2 \geq |x_1|\}$.

(ii) $K = \{(x_1, x_2) \in \mathbf{R}^2 : x_1 = 0\} \cup \{(x_1, x_2) \in \mathbf{R}^2 : x_2 = 0\}$.

(iii) $K = \{(x_1, x_2, x_3) \in \mathbf{R}^3 \mid x_1 \geq x_2 \geq x_3 \geq 0\}$.

(iv) $K = \{A \in S^{n \times n} : x^T A x \geq 0, \ \forall\, x \geq 0\}$.

Is any of the above cone a self dual cone?

14.4 *Show that $K^\star$ is a closed convex cone. Is $K^\star$ always pointed? Justify your reasoning.*

14.5 *Let $K_1 \subseteq K_2$. Prove that $K_2^\star \subseteq K_1^\star$.*

14.6 *Let $f(x, y) = \dfrac{x^2}{y}$, $x \in \mathbf{R}$, $y \in \mathbf{R}_+$ $y \neq 0$. Use the Schur's complement to write the inequality $f(x, y) \leq t$ as the linear matrix inequality (LMI).*

14.7 *Let A be an $m \times n$ matrix, $b \in \mathbf{R}^m$, $e \in \mathbf{R}^n$, $d \in \mathbf{R}$, $x \in \mathbf{R}^n$. Express the second order inequality*

$$\| Ax + b \|_2 \leq e^T x + d$$

as the linear matrix inequality (LMI).

14.8 *Write the equivalent SDPP of the following nonlinear programming problems*

(i) Min $4x_1^2 + x_1 x_2 + 4x_2^2$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 4 \\ x_1 &\leq 5. \end{aligned}$$

(ii) *Min* x_1
subject to

$$\begin{aligned} x_1^2 - x_2 &\leq 1 \\ -x_1 + x_2^2 &\leq \tfrac{1}{2}. \end{aligned}$$

(iii) *Min* $4x_1 + 3x_2$
subject to

$$\begin{aligned} x_1 + x_2 &\leq 3 \\ x_1 x_2 &\leq 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

14.9 *Prove that the condition*

$$\begin{pmatrix} \gamma - y - 1 & x \\ x & y + 1 \end{pmatrix} \succcurlyeq_{S_+^2} 0, \quad \forall\, y \in [0,1]$$

is satisfied whenever

$$\begin{pmatrix} \gamma - 1 & x \\ x & 1 \end{pmatrix} \succcurlyeq_{S_+^2} 0 \text{ and } \begin{pmatrix} \gamma - 2 & x \\ x & 2 \end{pmatrix} \succcurlyeq_{S_+^2} 0.$$

14.10 *Write the dual of the following SDPPs*

(i) *Min* x_1
subject to

$$\begin{pmatrix} x_1 - 1 & x_2 \\ x_2 & 1 \end{pmatrix} \geq_{S_+^2} 0$$

$$\begin{pmatrix} x_1 - 2 & x_2 \\ x_2 & 2 \end{pmatrix} \geq_{S_+^2} 0.$$

(ii) *Min* $x_1 + x_2$
subject to

$$\begin{pmatrix} 1 & x_1 - x_2 & 0 \\ x_1 - x_2 & x_1 & x_2 \\ 0 & x_2 & x_1 - 1 \end{pmatrix} \succcurlyeq_{S_+^3} 0.$$

15 Evolutionary Methods and Global Optimization

15.1 Introduction

Many engineering optimization problems contain multiple optimum solutions. Among them, one or more than one may be an absolute optimum or a globally optimal solution. Some of the search algorithms discussed in earlier chapters used information about the gradient and the Hessian of the functions involved in the optimization problem, to generate a locally optimal solution of the problem. Though gradient-based methods are computationally efficient, they typically provide a locally optimal solution and in general fail to provide the globally optimal solution to the problem. However, in many real life problems like combinatorial optimization, scheduling, reliability design, resource allocation, or portfolio optimization, one is ideally interested in finding the globally optimal solution. Often, functions of many variables have a large number of local optima, and searching for the global optimum is a difficult and intractable task. For instance, the six-hump camel back function given by

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 - x_1x_2 - 4x_2^2 + 4x_2^4, \quad x_i \in [-5,5] \ \ (i = 1,2),$$

has six local min points out of which only two are global min points $x_1^\star = (0.09, -0.713)^T$, $x_2^\star = (-0.09, 0.713)^T$ with $f(x^\star) = -1.0316$. The Rastrigin function,

$$f(x) = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i)), \quad x_i \in [-5.12, 5.12] \ \ (i = 1, \ldots, n),$$

where A is a positive integer, is a highly multi-modal function having only one global min point at $x^\star = 0 \in \mathbf{R}^n$ with $f(x^\star) = 0$.

If the problem in hand is a convex optimization problem, a local min point is guaranteed to be the global min point. Otherwise, for nonconvex problems (barring few exceptions, like minimizing a pseudoconvex objective function over a convex feasible set, etc.), a local min point need not be the global min point of the problem.

Each problem has its own distinguishing features that generally guide us to choose an appropriate optimization algorithm for determining its global optimum. One is therefore

required to have knowledge of the working principles of various optimization algorithms, their advantages and limitations. The primary objective of this chapter is to provide a brief description of a few popular methods that are widely used for global optimization. However, before we begin with our main aim, we would like to address the issue of *difficult problems*. It is these very problems that posed several challenges to algorithmic designers to construct algorithms that can efficiently solve the problems.

15.2 How Difficult is the Problem?

A little experience with optimization is enough to convince us that some problems are harder to solve than others. How difficult can a problem be? Are there problems for which no efficient algorithm is possible? These are some questions that scientists have been striving hard to answer. The concept of difficulty is related to the degree of sophistication of the algorithm. Roughly speaking, if one can not solve a problem or if it takes considerable time to solve it, the problem is termed as difficult. We briefly define below how to mathematically measure the difficulty of the problem.

An algorithmic problem is a mapping that yields a valid solution to the given problem or instance. An algorithm is a way to compute the mapping; it is a type of effective method which, given an initial state, a list of well-defined instructions for completing a task will proceed through a well-defined series of successive states that eventually terminate in an end-state. In general, the time taken to do so depends on the individual example. For all examples of a given size, we can determine the maximum time taken. This is the worst case complexity, and it is obviously a function of problem size. Different algorithms may complete the same task with a different set of instructions in less or more time or effort than others. The *worst case complexity* of an algorithmic problem is the time taken by the fastest algorithm that can solve the problem, as a function of the problem size N. For example, suppose we have three algorithms that can solve a given problem. The first one takes time that is a linear function of N, the second takes time quadratic in N, and the third takes time that is exponential in N. The worst case complexity of the problem is then linear.

It is easy to see that the actual time taken to solve an optimization problem depends on the machine being used, the quality of the code, and many such factors. These factors constitute implementation details. In order to allow a comparison that does not depend on implementation details, the notion of asymptotic analysis is used. *Asymptotic analysis* looks only at how the complexity grows with N, for sufficiently large N. Suppose that an algorithm takes $10N^2 + 5.6N + 25$ units of time. We can see that such an algorithm will take time that is bounded by $10.1N^2$. Since the actual constants involved are implementation dependent, it is more meaningful to examine how the time taken by an algorithm grows with N, for large problem sizes. It is clear that if another algorithm B takes $4N^3$ time in the worst case, for large N, then it would be slower than algorithm A that takes $10.1N^2$. We say that the worst case complexity for algorithm A is $O(N^2)$ while

that for B is $O(N^3)$. This notation eliminates the need for implementation dependent constants. It implies that

$\exists$ a positive integer N_0 and a real number K such that the worst case run time of the algorithm $A \leq KN^2$, $\forall\ N \geq N_0$.

The worst case complexity of an algorithmic problem is clearly the smallest or minimum across all algorithms available for the task. The worst case measure has some drawbacks. It is possible that the average case behavior may be much better and the worst case measure is overly pessimistic. However, in many domains it still serves as a valuable measure that help us understand how difficult some problems can be, in comparison to others.

An algorithm is termed as being a *polynomial time* one if its worst case run time complexity is $O(N^k)$ for some non-negative fixed real number k. Note that k does not depend on N. A polynomial time algorithmic problem is the one for which a polynomial time algorithm is available for solving it. The term feasible is also used in place of polynomial, since for large N, the time taken by a non-polynomial time algorithm would grow to be prohibitively large. The class of all problems that have feasible or polynomial time algorithms is denoted by 'P'.

On the other hand, an important class of algorithmic problems are *decision problems*, where the solution is simply a binary one, that can be treated as a yes/no, or true/false decision. For instance, consider a graph $G = (V, E)$ defined by V nodes and their adjacencies. The graph coloring problem involves assigning a label to each node such that no two adjacent nodes have the same color or label. The optimal coloring problem is to find an assignment with the minimum number of colors, whereas finding whether the given graph can have a coloring in 5 colors or less is a decision problem.

Many decision problems require us to use *randomized algorithms* to solve them. A *non-deterministic algorithm* is a randomized algorithm which employs some randomization as part of its logic. The set of all decision problems that have polynomial time non-deterministic algorithms is called 'NP'. For example, two layers channel routing in VLSI, scheduling problems, and a number of tasks in circuit design can be transformed to versions of the optimal coloring problem. All these tasks are in class NP, which includes a very large number of such problems arising in several domains of engineering. Some of the algorithmic problems share a property that every problem in NP can be reduced to them in polynomial time. These are special NP problems and are termed as 'NP- complete' problems. Examples of such problems include optimal coloring of graphs, the traveling salesman problem, and the knapsack problem. For many algorithmic optimization problems, involving complicated outputs, the threshold problem is NP-complete. Such problems are termed as 'NP-hard' problems. We here skip the details, but the readers can refer to [125].

It is clear that for several problems, not only is it hard to obtain the globally optimal solution, but it is also hard to determine how close we are to the global optimum. The only polynomial time algorithms for many such problems are non-deterministic

randomized algorithms with a clairvoyant random number generator. While such algorithms are impractical to implement, it indicates that randomized algorithms may be the only feasible way to search for good solutions to such hard problems.

The above discussion on algorithmic complexity indicates that deterministic approaches used for finding the globally optimal solutions may take a very long time, particularly for real world problems that involve several variables. Most global optimization methods that are widely applied therefore involve some randomized search steps, or are 'heuristic' in nature.

Algorithms that can be used to define heuristic methods, and are applicable to a wide set of different optimization problems are called 'metaheuristics'. Examples of metaheuristics include 'simulated annealing', 'genetic algorithms', 'ant colony optimization', 'particle swarm optimization', and 'tabu search', to name a few. The idea is to leave the best solution untouched while allowing other states to explore the search space. These approaches also try to combine features of several solutions in an attempt to find the best solution. The use of metaheuristics has seen a significant increase in last decade because of their ability to find high quality solutions to otherwise hard combinatorial optimization problems. In the sequel to follow, we describe some of these metaheuristics.

15.3 Simulated Annealing

Simulated Annealing (SA) is a randomized search technique in numerical optimization based on the principles of thermodynamics. The idea of SA comes from a paper published by Nicholas Metropolis et al. [115] in 1953. Annealing, in metallurgy, is a technique that involves extensive heating followed by a controlled cooling of a metal. At high temperature, the atoms in the metal are in a highly disorganized liquid state. When the temperature is gradually lowered, according to a cooling schedule, the atoms in a metal organize themselves into a highly ordered solid state. The solid structure is a stable configuration corresponding to a lower energy state. The structural properties of the solid depend on the rate of cooling. SA exploits this analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure.

While solving an optimization problem, it is experienced that the search tends to get stuck in a local min point from which there is no escape route, and very often this point is not a global min point of the problem. SA helps us to design a possible technique to overcome this hurdle by allowing the point to bounce over mountains and traverse new valleys. To visualize this phenomena, we imagine the objective function of an optimization problem as a geographical terrain, as shown in Fig 15.1.

In a crystal lattice, atoms vibrate or shake about their mean positions. The higher the temperature, the more vigorous the shaking. We now consider how a simulated system of atoms can help us find the deepest valley (or global minimum) in this terrain.

Suppose that the current state corresponds to a local min point such as P_1. The SA process begins at a high temperature; which implies that the atoms are shaking

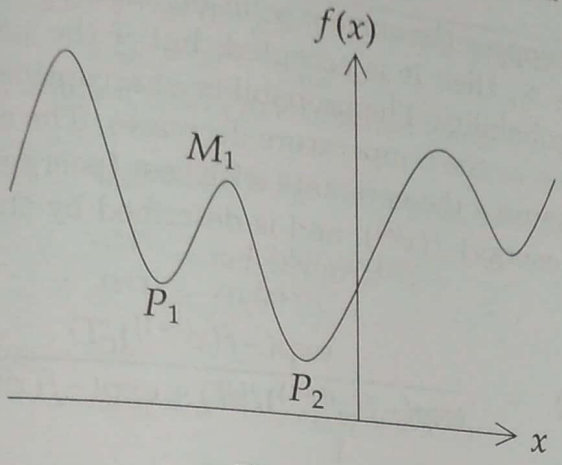


Fig. 15.1.

vigorously, and can collectively move to a much higher energy state, such as M_1. As the temperature is reduced, the shaking reduces, and the search becomes more local, i.e. moves to higher energy states become less likely. In time, the system moves top state P_2.

The SA algorithm involves generating a new state and accepting it by applying an acceptance criterion. New states are usually generated by applying a set of 'moves' or transformation rules to the present state. Generated states may have a cost or energy that is occasionally higher than the present state.

the acceptance criterion accepts the new state, i.e. a transition is made to it from the present one, if its energy or cost is higher than that of the present state, it is accepted with some probability. The probability of acceptance depends on the difference in energies as well as a global parameter T, called temperature.

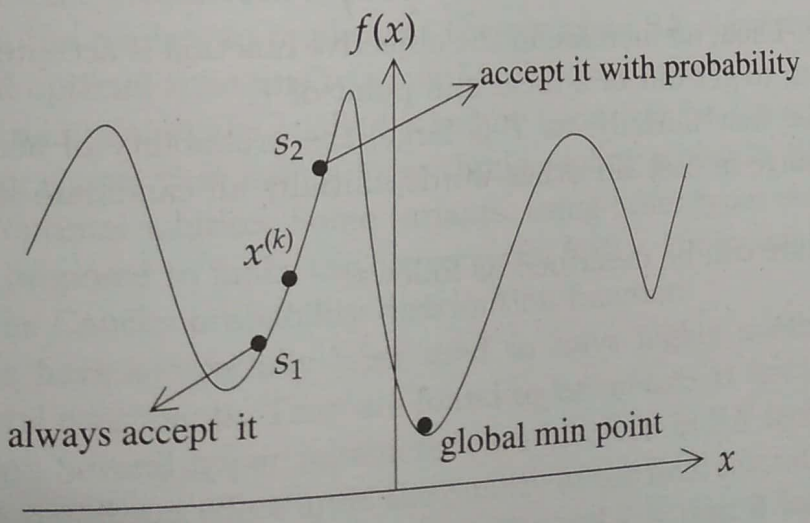


Fig. 15.2.

Referring to Fig 15.2, suppose the current solution (state) of an optimization problem is $x^{(k)}$. If the new state is s_1, then it is accepted; but if the new state is s_2, it is only accepted with a certain probability. The probability of accepting a worse state is high at the beginning and decreases as the temperature decreases. The acceptance probability is based on the chance of obtaining the new state with cost (energy) $f(x^{(k+1)})$ relative to the previous state with cost (energy) $f(x^{(k)})$, and is described by the *Boltzmann probability distribution function*, i.e.

$$\begin{aligned} p((\Delta f)_k) &= \frac{\exp(-f(x^{(k+1)})/T)}{\exp(-f(x^{(k+1)})/bT) + \exp(-f(x^{(k)})/T)} \\ &= \frac{1}{1+\exp((\Delta f)_k/bT)} \\ &\simeq \exp((-\Delta f)_k/bT), \end{aligned}$$

where $(\Delta f)_k$ represents the difference between the present and previous values of the costs (energies), i.e. $(\Delta f)_k = f(x^{(k+1)}) - f(x^{(k)})$, T is the temperature, and b is the Boltzmann's constant that is used to normalize the energy function values. For practical purposes, b is taken to be 1.

The search for the minimum is initiated at a random feasible point $x^{(0)}$. Set $x_{\min} = x^{(0)}$, $f_{\min} = f(x^{(0)})$. We start with an initial state of temperature $T = T^0$ which is set to a high level.

The change in a state from the present state to the candidate new state is accepted if

(i) $(\Delta f)_k < 0$, i.e. the function value is decreased. This forces the system towards a state corresponding to a local or a possibly global min point.

(ii) If $(\Delta f)_k > 0$, but $p((\Delta f)_k) = \exp\left(\frac{(-\Delta f)_k}{T}\right) > r$, where $r \in (0,1)$ is a randomly generated number. Then, an increase in the objective function is accepted with a certain probability in order to get out of a local min point of f.

It is easy to see that initially, as T is large, the probability of acceptance is high for almost any change in cost. In other words, initially all candidate states tend to be accepted.

The SA procedure can be described as follows.

$T = T^0 \gg 0$, $x = x^{(0)}$, $e^{(0)} = f(x^{(0)})$;
$x_{\min} = x^{(0)}$, $e_{\min} = e$;
Set objective function (energy) evaluation count $k = 0$;

While $k < k_{\max}$ *and* $e > e_{lb}$ (stopping criteria) (15.1)

$x^{(k+1)} = neighbour(x^{(k)})$,
$e^{(k+1)} = f(x^{(k+1)})$;

If $e^{(k+1)} < e_{\min}$, *then*
$x_{\min} = x^{(k+1)}$, $e_{\min} = e^{(k+1)}$;

else

if $p((\Delta e)_k) > random(\cdot)$, *then*
$x_{\min} = x^{(k+1)}$, $e_{\min} = e^{(k+1)}$;

else
$x_{\min} = x^{(k)}$, $e_{\min} = e^{(k)}$.

Decrease the temperature T.
Set $k = k + 1$, *and go to* (15.1).

return $x_{\min}$.

Here, $k_{\max}$ is the maximum number of iterations allowed, and e_{lb} is the pre-targeted lower bound on the objective function $f(x)$. Also, the temperature is usually lowered in a geometric progression, i.e. $T^{(k+1)} = \beta(T^{(k)})T^{(k)}$. Here, $\beta(T^{(k)})$ is a function that indicates the rate of change, and it is termed as the *cooling schedule*. The choice of function $\beta(\cdot)$ is critical for the efficiency of the algorithm, and much efforts has been devoted to optimizing it for some commercial applications.

We encourage the readers to implement the complete SA algorithm and test run it to find the global optimal solutions to some *benchmark problems*.

One needs to realize that the cooling schedule is critical for the success of the algorithm. It has been shown that with a logarithmic cooling schedule, SA, asymptotically converges to the optimal solution. Some variants, using other types of distribution functions, have been proposed to make the convergence faster. For example, 'fast simulated annealing' uses the Cauchy probability distribution function.

SA algorithms have successfully been used to solve highly nonlinear optimization models with several constraints. They are found to be *robust*. At the same time, SA has certain weaknesses. Several hyper parameters are involved in SA techniques, and their tuning can have a significant effect upon the quality of the final solution. Another crucial factor is time. In fact one of the major drawbacks of SA is that it can be prohibitively slow on high dimensional optimization problems.

It is clear from Fig 15.2 that the choice of the initial point can have a large bearing on the efficiency of search. A logical extension of this observation is to commence the search from different initial locations - this is known as *restart*. Alternatively, one could begin from several different locations in a *multistart approach*. Since the different local optima can probably capture a few different features of global optimum, by combining the features of different local optima, it is natural to believe that the search for the global optimum will become faster. This approach has been successfully used in 'evolutionary optimization methods'. Genetic algorithms are a widely used technique from this genre.

15.4 Genetic Algorithms

Genetic algorithms (GA) find their motivation from Charles Darwin's famous theory of natural selection. This embodies a widely held notion that all life are related and has descended from a common ancestor. GAs in particular became popular through the work of John Holland [77] in early 70's.

The algorithm begins with a set of solutions called a *population*. Solutions from one population are taken to form the new one with the hope that the new population will be better than its ancestor one. The algorithm thus tries to simulate the evolution of a species in which a population of individuals, over successive generations, optimizes a set of traits or characteristics that maximizes the *fitness* of individuals in a given environment. This is achieved by assigning a numeric fitness value to each individual in the population. For each generation individuals are selected from population for 'reproduction', 'crossover' and 'mutation', to give birth to new individuals. Selection of the next generation population is entirely based upon the fittest individuals from the *parent* and the newly formed *offspring* generations. The idea is that offspring of fit parents would inherit good traits or features of their parents and some of the offspring would inherit a better combination of traits making them even fitter. This strategy allows an optimization method to explore the *search space* in a clever way. The fitness of the individuals is expected to improve over time and the best individual is chosen as a solution after several generations.

GA's use two basic processes:
(i) passing over features from one generation to the next generation;
(ii) survival of the fittest.

We now explain the terminology and the procedure associated with a basic GA.
Population. The set of several alternate solutions of the given optimization problem.
Chromosome. Each individual in the population.
Gene. The chromosomes are mathematically coded as a string, with each character or symbol in the string being called gene. Each gene encodes a trait, for example, the color of skin, height, potential to be afflicted with a particular disease, etc.
Fitness Function. An evolution function used for determining the fitness of each chromosome. This function is generally user defined and problem specific.

Search Space. The set of all feasible solutions of the given problem.

Encoding Schemes. In order to apply GA's to solve a given optimization problem, one needs to encode the chromosome appropriately. In fact the chromosome in some way must contain information about the solution which it represents. Appropriate encoding of the chromosome is vital and it is heavily dependent on the given problem.

The most popular and widely used way of encoding is *binary coding*. In this coding, every chromosome is a string of bits (0 or 1). An encoded chromosome might look like 1101101.

For example, suppose an optimization problem involves a bounded variable x with $x^L \le x \le x^U$. If 5-bits are used to code each variable, then the string 00000 represents x^L, and the string 11111 represents x^U. The following linear mapping is usually used to code other values of x in (x^L, x^U) as a substring s of length l-bits,

$$x = x^L + (\text{decoded value of } s)\left(\frac{x^U - x^L}{2^l - 1}\right). \tag{15.2}$$

Suppose $x \in [0, \pi]$, and 5-bits are used for coding. Then $x = 0$ is coded as 00000 and $x = \pi$ is coded as 11111. Now, if $s = 01011$ then its decoded value is $1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 1 * 2^3 + 0 * 2^4 = 11$. The corresponding value of x is calculated using (15.2), and it is equal to $0 + 11\left(\frac{\pi - 0}{2^5 - 1}\right) = 1.1148$.

'Permutation encoding' is another popular coding scheme that have been used in ordering problems, such as traveling salesman problem or task ordering problem. In permutation encoding, every chromosome is a string of numbers in a sequence. The encoded chromosome looks like, 153264798. This encoding is useful only for ordering problems.

There are some other useful encoding schemes, like, 'value encoding' and 'tree encoding'. These schemes have been successfully used in some very specific optimization problems.

Fitness Function. A fitness function $\mathcal{F}(x)$ is generally derived from the objective function $f(x)$. For maximization problem, the fitness function is usually taken as the objective function itself, i.e. $\mathcal{F}(x) = f(x)$, whereas for the minimization problem, the fitness function is often taken as $\mathcal{F}(x) = \dfrac{1}{1 + f(x)}$.

For instance, suppose we want to maximize $\cos(x)$, $x \in [0, \pi]$. Suppose 5-bits coding is used, and one of the coded chromosome is $s = 01011$. The corresponding point is $x = 1.1148$ (described above). The fitness value of x (or s) is equal to $\mathcal{F}(x) = \cos(1.1148) = 0.44036$.

The next step in GA involves selecting the initial population of chromosomes. The size of the population is vital. If a very small population size is chosen, then there will not be enough exploration of the search space, while if the population size is too large, then it may take a long time to find the fittest chromosome. Numerical experiments have indicated that a population size of $20 - 30$ is reasonable for many optimization

problems, but for some problems, a population size of $50-100$ has been shown to work well.

After the initial population is randomly generated, the algorithm evolves through GA operators. There is no dearth of possible operators. However, commonly employed ones are 'selection or reproduction', 'crossover', 'mutation', and 'inversion'.

The 'selection operator' selects good strings in a population and forms a mating pool. The basic idea is that the above-average strings are picked from the current population and their multiple copies are inserted in the mating pool with a probability proportional to its fitness. If n is the population size, then the probability for selecting the i-th string is

$$p_i = \frac{\mathcal{F}_i}{\sum_{i=1}^{n} \mathcal{F}_j} \,. \tag{15.3}$$

The string with a higher fitness value has a higher probability of being copied in the mating pool.

Example 15.4.1 *Generate a mating pool for the problem: Max* $3x - x^2$ *over* $[0,3]$.

Solution Let 5-bits binary coding scheme is used, and suppose the population size is $n = 4$ (this is only for illustration purpose), and it is constituted of $\{(01001), (10100), (00001), (11010)\}$. The fitness function is taken as the objective function itself, i.e. $\mathcal{F}(x) = 3x - x^2$. The population after selection, constituting the mating pool, is depicted in the following table. The column p_s is computed using (15.3).

string s	decoded s	x	$\mathcal{F}(x)$	p_s	mating pool
01001	9	0.8710	1.8544	1.3703	01001
10100	20	1.9355	2.0603	1.5224	10100
00001	1	0.0968	0.2810	0.2076	10100
11010	26	2.5161	1.2175	0.8996	11010

The 'crossover operator' is the main operator. It combines a part of the chromosome of one parent with a part of the chromosome of the other parent. A simple way is to split each parents chromosome at a cut point and choose the left half of the chromosome of the first parent and the second half of the chromosome of the other parent. For example, suppose we have two parents chromosomes, Chromosome 1 and Chromosome 2, and suppose we select the crossover site after third gene of the two chromosomes. Two offsprings, namely Offspring 1 and Offspring 2, are produced as a result of crossover. Offspring 1 is formed by taking the left part of Chromosome 1 and the right part of Chromosome 2 while Offspring 2 results from the combination of the left part of Chromosome 2 and the right part of Chromosome 1.

Chromosome 1 : $\overbrace{110}\,|\underbrace{00100}$; Chromosome 2 : $\underbrace{110}\,|\overbrace{11000}$

Offspring 1 : $\overbrace{11011000}$; Offspring 2 : $\underbrace{11000100}$.

The above is an illustration of a *one point crossover*. However, the *two point crossover* is also frequently used in the GA algorithms. The following illustrates the two point crossover, sites selected after the third and fifth genes.

Chromosome 1 : $\overbrace{110}\,|\,\underbrace{00}\,|\,\overbrace{110}$; Chromosome 2 : $\underbrace{110}\,|\,\overbrace{11}\,|\,\underbrace{011}$

Offspring 1 : $\overbrace{11011110}$; Offspring 2 : $\underbrace{11000011}$.

Crossover may be applied many times to generate several offsprings from each set of parents. The ratio of the number of offspring to the population size is termed as the *crossover rate or crossover probability*. If the crossover probability is 100%, then all the offspring in the next generation are obtained by crossover, while if the crossover probability is 0%, it indicates that the whole new generation is formed from entirely exact copies of chromosomes of the old population. Therefore, a positive crossover probability is desirable so that the chromosomes in the next generation contain good traits of the parents chromosomes, with some additional new features. At the same time, it is good to have a crossover probability less than 100%, so that some fit parents survive to the next generation, and good traits are retained in a stable fashion.

After the crossover is performed, we modify the population in the next generation by replacing the parents in the older population by their offsprings. The population size is maintained a constant throughout the algorithm.

Next, we apply the 'mutation' operation. In nature, mutation is refereed to the random changes in the genetic constitution of cells. It is considered to be responsible for the generation of new species, new traits, and often the source of critical adaptations that have allowed a species to survive hostile changes in their environments. It is believed that mutation is responsible for many bacterial species developing resistance to antibiotics.

In genetic algorithms, mutation is simulated by randomly changing some of the string elements. The simplest way of achieving this is by randomly interchanging genes, i.e. for binary coding, switch a few randomly chosen bits from 1 to 0 or from 0 to 1. For example, if the original offspring is 110110, then the mutated offspring can be 010111, which is obtained after switching the first gene from 0 to 1 and the sixth gene from 1 to 0.

Since crossover combines chromosome fragments from parents, some genes may disappear from the chromosomes of the parents after many generations of crossover. Mutation helps to re-introduce these missing genes and thus ensures that the search does not get

trapped in local minima. For instance, suppose that all the chromosomes in the current population have the first gene code 0, while the global min point has the first gene code as 1. No matter how long we apply the crossover operation alone, the algorithm will not be able to produce a chromosome with the first gene code as 1. However, the same can be obtained through the use of the mutation operator in a single iteration of the algorithm.

The *mutation rate or mutation probability* is the ratio of the number of random gene changes to the total number of genes in a population. Generally, mutation probability is low, otherwise the algorithm becomes more of a random search.

There is no general theory available to tune the crossover and the mutation probabilities. Empirical studies suggest that the crossover probability should be high, somewhere between 80% – 95%, while the mutation probability should be low, about 0.5% – 1%.

Another operator, namely the 'inversion' operator, involves taking a random substring from an offspring and inverting it end-to-end. Inversion is not applicable in all scenarios. It is useful when the chromosomal representation depends only on the set of genes and not on their sequence, i.e. when the position of a gene is not important but only whether it is present or not. Inversion is rarely used in applications and its benefits are somewhat unclear.

The GA procedure is summarizes as follows.

Step 1. Choose a coding to represent problem parameters; a selection operator; a crossover operator; a mutation operator; population size; crossover probability; mutation probability; maximum allowable generation number $k_{\max}$.

Step 2. Set the counter $k = 0$. Evaluate the fitness of each string in the population.

Step 3. If $k > k_{\max}$, terminate the algorithm.

Step 4. Perform selection, followed by crossover, thereafter mutation.

Step 5. Evaluate the fitness of the newly generated chromosomes.

Step 6. Determine the next generation. This could be composed of newly generated chromosomes or a mix of individuals from the previous set and the newly generated set. The fitness value of a chromosome is used to select it from the entire population set. Generally speaking, it is desirable not to select only the fittest individuals but to have some diversity in the population so that exploration of the search space continues.

GA and its variants have many success stories to tell. They are capable of handling a fairly large class of optimization problems, including those where the objective function is highly non-smooth. The list of applications of GA, in many problems of sciences, engineering and management, is fairly extensive. However, GAs also have their own share of disadvantages. For some optimization problems and instances, simpler optimization algorithms are found to outperform GAs (given the same amount of computation time). The implementation and evaluation of the fitness function is an important factor in the speed and efficiency of the algorithm. GAs cannot effectively solve problems in which the only fitness measure is right/wrong. Moreover, GAs cannot sacrifice short-term fit-

ness to gain longer-term fitness. The latter sometimes causes GAs to converge towards locally optimal solutions, or even arbitrary points rather than the global optimum of the problem. GAs fail to adapt to sudden dynamic changes in the problem in real time. By dynamic changes in the problem, we mean that the problem definition changes during run time, e.g. a network routing problem where the data can vary with time. Consequently, the genetic evolution process becomes slow and may takes many generations in practice. In such cases, GAs involve prohibitive amount of computations.

15.5 Ant Colony Optimization

There are other examples in nature from where one can draw inspiration for efficient forms of search. Insect colonies are good examples. How do social insects achieve self-organization? Bees not only manage to find food sources efficiently by employing a distributed search, but also employ elaborate *signalling schemes* (a kind of intelligence system) to help other members of the colony relocate the same source. Bacterial colonies are now known to use chemical signalling to help develop resistance to drugs, or to locate food sources. An easy to observe species is the common garden ant. An ant looking for food lays down a trail of a chemical called a *pheromone*, as it forages. Pheromones are chemicals that trigger a natural behavioral response in another member of the same species. Other ants are more likely to follow trails which have more pheromone deposits. This indirect communication with the environment helps the ants to achieve some form of self-organization in their behavior. This process of engineered self-organization is termed as *stigmergy* - derived from the Greek words *stigma* and *ergon* meaning sign and action, respectively. Stigmergy is an indirect form of stimulation between two individuals, where one of them modifies the environment and the other responds to the new environment at a later time. Examples include termites or wasps building their nest, or ants following a trail.

Ant colony algorithms attempt to simulate the process of stigmergy. These algorithms exploit an *artificial stigmergy*, as a means of coordinating *artificial ants*, to obtain solutions to optimization problems. One of the most popular ant colony algorithm is called *ant colony optimization* (ACO), which is used to solve discrete optimization problems. The ACO algorithm was introduced by Marco Dorigo in the 90's, (he was awarded the *Marie Curie research excellence award* by the European Commission in 2003 for his contributions to metaheuristics). The ACO algorithm and its variants are inspired by the behavior of ants in finding paths from their colony nest to food.

Consider Fig 15.3, which shows two paths from the nest to the food source, where one path is longer than the other one.

Figure 15.3 illustrates an experiment carried out by Deneubourg et al. [47], who studied foraging by an Argentine ant species. This species uses a trail pheromone to mark paths to food. At the start of the experiment, there is no pheromone on either path. Thus, the ants (both real and virtual ones) choose both paths with equal probability.

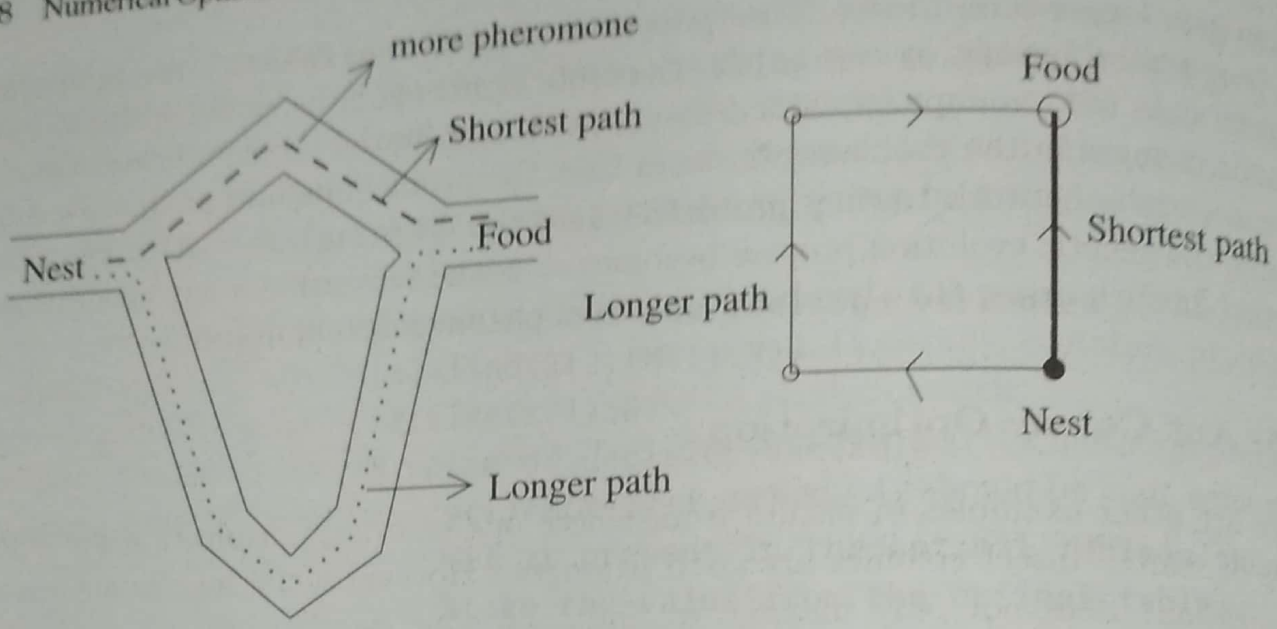


Fig. 15.3.

While walking down the paths, ants deposit pheromone. The following ants prefer paths that have more pheromone. This leads ants to choose the path that has more pheromone deposit. Ants that reach the food source early, using the shorter path, return to the nest earlier. Therefore pheromone starts to accumulate faster on the shorter path. Eventually, the pheromone trail laying mechanism induces a positive feedback that rapidly increases ants' bias towards the shorter route. After some time, it is observed, that nearly all the ants prefer the shorter path that also has an increasing amount of pheromone. In this model, ants deposit pheromone both on their forward path from the nest to the food source, and the return path to the nest. It was experimentally observed that if it is assumed that the ants deposit pheromone one way, either only on their forward trip or only on their return trip, then the ant colony is unable to choose the shortest path.

In real-life scenario, one can easily relate the ACO algorithm with finding the shortest path between two nodes in an undirected graph. Artificial ants (replica of real ants for problem solving) are used to trace the shortest path or the minimum cost path in the given graph by mimicking the behavior of real ants. However the ants quickly fail because they start traveling in *loops*. The looping phenomenon is easy to explain. Once an ant travels in a loop it starts reinforcing the pheromone trail rapidly and the loop becomes the preferred path for other ants to follow. One measure to overcome this hurdle is to do the pheromone update in the algorithm only in the return mode, and completely remove it from the forward mode, i.e. assuming that the ants do not leave the trail of pheromone on their forward journey but deposit pheromone only on their return journey. However, in this case the algorithm fails to converge. This fact has also been mentioned above. Thus one needs to introduce some modifications in the behavior of artificial ants.

One way is to provide artificial ants with a 'limited form of memory' to store partial information regarding paths traversed earlier and the link costs. The memory is enough

for the artificial ants to (i) construct probabilistic forward paths; (ii) retrace the path that they had constructed in a deterministic way on return journey along with loop eliminations; (iii) evaluation of solution quality.

Another factor that has been incorporated in some ant colony algorithms is a finite rate of *evaporation* of the pheromone. This enables ants to avoid repeated traversal of the paths that they had already visited, and allows them to pick alternative routes.

We now describe a 'simple ant colony optimization (S-ACO)' algorithm for finding the shortest path between two nodes S and D in a graph $G = (V, E)$. The graph G has V number of vertices and E number of arcs or edges.

Two nodes $i,\ j \in V$ are said to be 'neighbors' if there exists an arc $(i, j) \in E$. The neighborhood concept can be depicted by the edge-weights defined as follows.

$$e_{ij} = \begin{cases} 1, & \text{if there is an edge from node j to node i,} \\ 0, & \text{otherwise.} \end{cases}$$

The amount of pheromone on the arc between nodes i and j is denoted by q_{ij}. *We assume that the ants do not deposit any pheromone while moving in the forward path.*

Let the number of ants be N; let the source (nest) node be denoted by $S \in V$ and the destination (food) node be denoted by $D \in V$. Also, we shall be using $pred(k, i)$ to denote the node visited by the ant k just before visiting the node i.

The algorithm is designed so that the amount of pheromone on a path is a monotonic function of the *goodness of that path*. The goodness of the path is estimated by the foraging behavior of the ants.

The general framework of the algorithm may be summarized as follows.

Step 1. *(Initialization)* Set $q_{ij} = q_0,\ \forall\ i,\ j$. This ensures that all edges in G have an equal non-zero amount of pheromone on them.

Step 2. *(Forward Path Construction / Forward Mode)* Suppose the r-th artificial ant is currently at the i-th node in G. The forward path, from S to D, is build by probabilistically choosing the next node to move to among those that are in a neighborhood of the current node i in G. The probabilistic choice is governed by the amount of pheromone previously deposited on G by the earlier ants $(1, \ldots, r-1)$ that had traversed those nodes during their return journey. Remember, ants in the forward mode do not deposit any pheromone on G. The probabilities of movement are computed as follows.

For each ant $k = 1, \ldots, N$,

For each node $i = 1, \ldots, V$,

compute the probability of an ant moving to the next node j as

$$prob(k, i, j) = \begin{cases} e_{ij} \dfrac{q_{ij}^{\alpha}}{\sum_{k=1}^{V} q_{ij}^{\alpha}}, & j \notin pred(k, i), \\ 0, & \text{otherwise,} \end{cases}$$

where the parameter α defines the relative influence of the pheromone level q_{ij}.

Move each ant to a new node based on the above probabilities. Note that if the node j is not in a neighborhood of node i in G, then $e_{ij} = 0$, and thus $prob(k, i, j) = 0$.

On the other hand if the graph G is a *weighted graph* with edge - weights representing the distance d_{ij}, then the probabilities are computed as follows.

$$prob(k,i,j) = \begin{cases} e_{ij}\dfrac{q_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{k=1}^{V} q_{ij}^{\alpha}\eta_{ij}^{\beta}}, & j \notin pred(k,i), \\ 0, & \text{otherwise}, \end{cases}$$

where η_{ij} is the *heuristic information* denoting the desirability of the arc (i, j), and it is typically taken as $\dfrac{1}{d_{ij}}$; also β is a parameter to control the influence of η_{ij}.

The ant memorizes the nodes it had visited and the costs of the edges (edge lengths d_{ij}) traversed. The cost of the solution generated by the ant can thus be evaluated and stored in the memory. This memory is used later during the return journey.

Step 3. *(Return / Backtrace Mode with pheromone update)* The return journey follows the deterministic rules that are described in the following (i)-(iii) points.

(i) Once the r-th ant reaches node D, it switches its gear from the forward mode to the return mode. It retraced the same path that it had build in its forward journey, step by step, till it reaches the starting node S. Remember the ant had already memorized the forward path. An additional feature of the return journey is the elimination of any loop formed during the forward phase.

Loop elimination is carried out in a systematic iterative manner through path scanning scheme. In the present text we are skipping the description of the loop elimination procedure, but interested readers are encouraged to refer to an excellent text by Dorigo and Stützle [51] for details.

(ii) On its return trip to node S, the r-th ant deposits some amount of pheromone on the edges of G that it has visited. If the amount of pheromone deposited on the return path is a constant, then $q_{ij} = q_{ij} + \Delta q^r$. However, the cost of the solution (also called a quality of the solution) generated in the forward journey of the r-th ant can also be used to define a non-increasing function of the path length, and that can be used for pheromone updating. The latter function represents an appropriate choice for updating pheromone as the ant deposits the pheromone earlier on the short path, consequently, the pheromone deposit along shorter edge lengths is more as compared to the pheromone deposit along the longer edge lengths. This deterministic strategy becomes quickly biased toward the shorter paths between the nodes S and D in G.

(iii) A phenomena similar to evaporation of pheromone is added in the algorithm. This is done by allowing the *forgetting* of poor choices done in the past. The algorithm is allowed to progressively 'learn' from the past mistakes. This way an element of 'artificial intelligence' is incorporated in the algorithm. The evaporation process, in turns, helps

to discover alternative routes from S to D over the period of time. The pheromone trail is evaporated by applying the following rule

$$q_{ij} = (1 - \rho)\, q_{ij}\,, \quad \forall\, (i, j) \in E,$$

where $\rho \in (0, 1]$ is an *evaporation rate constant* or *memory loss*.

The standard procedure in S-ACO is to first apply the evaporation rule, and thereafter let the ants deposit the pheromone on their return trip.

Several variants of S-ACO have been proposed to help escape from the local min point, to make the search faster, and to avoid solutions that have been found in earlier iterations. ACO algorithms have been tested on large number of problems and are found to be phenomenal successful. The readers are referred to the wealth of literature on ant colony optimization, some of which is mentioned in the Summary of the chapter.

15.6 Summary and Additional Notes

- The search for the global optimum is akin to the search for the truth or the holy grail, and there are as many paths to it as the number of faiths. But, like in real life, every path consists of some smooth portions that can be seen as advantages, and many hurdles that can be visualize as drawbacks. One has to thrive hard to achieve success. The success, many a times, is goal dependent, and there are no shortcuts to it. This can be interpreted as the choice of an appropriate technique to obtain the global optimal solution may depends on the specific problem in hand. The focus in this chapter has been to familiarize the readers with some widely applied techniques.
- After briefly describing the meaning of difficult optimization problems in Section 15.2, we discussed the working of the three popular heuristics in Sections 15.3 to 15.5.
- One of the major advantages of these heuristics is their gradient-free nature. In other words, it means, that these algorithms can be applied to a wide class of optimization problems, which involve highly non-smooth functions. Consequently these algorithms have been applied to successfully solve many hard core combinatorial optimization problems, and problems relating to multiobjective optimization, subset selection, neural network, scheduling, routing in networks, molecular structure, mobile communications infrastructure, and machine learning, to name a few. There are many other salient features of these metaheuristics, such as robustness, efficiency and fast convergence. They generally yield extremely good solutions, usually close to the global optimum.
- We list a few links relevant to the techniques described in the chapter.
For SA, refer to Lester Ingbers homepage - http://www.ingber.com/
- ASA, CalTech code in C and a Matlab Interface;
'General Simulated Annealing Algorithm', an open-source MATLAB program -

http://www.mathworks.com/matlabcentral/fileexchange/; or
http://www.heatonresearch.com/articles/64/page1.html; and
http://paradiseo.gforge.inria.fr/ also provide open-source SA files.

For implementations of GA, refer to:
GENOCOP - http://geneura.ugr.es/~jmerelo/EO.html;
GAlib - http://lancet.mit.edu/ga/;
GOAL - http://www.geocities.com/geneticoptimization/;
GAGS - http://kal-el.ugr.es/GAGS/;
Genetic Algorithm and Direct Search Toolbox in MATLAB.

The best source of ACO implementations is:
http://www.aco-metaheuristic.org/aco-code/public-software.html.
Another interesting site -
http://www.codeproject.com/KB/recipes/GeneticandAntAlgorithms.aspx
contains programs for solving the traveling salesman problem by ACO and GA using MATLAB.

- The years after 1990 have witnessed tremendous growth in metaheuristic algorithms. Besides the three metaheuristics described in this chapter, a number of other metaheuristics are available in literature. Some of the popular ones are 'particle swarm optimization' (PSO), developed by J. Kennedy and R. C. Eberhart (http://www.particleswarm.info/); 'tabu search' attributed to F. Glover (http://spot.colorado.edu/ glover 'harmony search'(HS), mimicking the improvisation process of musicians (http://www.hydroteq.com); 'greedy randomized adaptive search procedure' (GRASP) by T. A. Feo and M. G. C. Resende.
- Besides of course the aforementioned web links, there are several excellent books dealing exclusively with specific metaheuristics. The reader may like to refer to the texts by Laarhoven and Aarts [99] for SA; M. Mitchell [116] for GA; Dorigo and Stützle [51] for ACO; M. Clerc [38] for PSO; Glover and Laguna [69] for tabu search; and the text by X. Yang [170].
- Innumerable variants and hybrids of these techniques have been proposed, and many more applications of metaheuristics have been reported. This is a field of active research, with a large community of researchers and users, and a wide range of applications.

16 Mathematical Programming Applications in Machine Learning

16.1 Introduction

The term *Machine Learning* refers to learning from empirical data which could be in the form of images, measurements, observations, patterns, or records. The ultimate goal of any machine learning algorithm is to perform well on the training data and also to ensure a good performance on future, previously unseen data. Therefore, learning in this context means an inductive process where one observes examples that represent incomplete information about some *statistical phenomenon*. Three basic problems that are often studied in machine learning are *classification, clustering and regression*. Although there are several approaches available in the machine learning literature to address these basic problems, we concentrate here on the mathematical programming approach only. This approach has been initiated by Mangasarian [110] and has now become very popular in the machine learning community as well as in the mathematical programming community. There is a very close connection between machine learning and mathematical programming, because most of the models for classification, regression and clustering result into linear programming, quadratic programming or certain specialized convex programming problems e.g. semi-definite programming and second order cone programming. An added advantage of studying machine learning problems via mathematical programming is the fact that it gives an access to all nice theoretical results (e.g. KKT conditions and duality theory) and efficient algorithms available in mathematical programming so as to use them for the specific problem at hand.

Machine Learning techniques have been successfully applied to a wide variety of areas, e.g. bio-informatics, computer vision, financial forecasting, network intrusion detection, spam categorization and text categorization. These techniques can broadly be classified into two classes, namely, *supervised learning* and *unsupervised learning*. The problems of classification and regression belong to the supervised learning class whereas clustering is in the class of unsupervised learning.

In classification and regression every sample is associated with a label. If the labels are discrete, then the task is termed as a classification problem, otherwise the problem

becomes a regression one for real valued labels. Based on the labeled training examples, one is particularly interested in predicting the label for newer samples. Hence, learning is not only a question of finding the relation between samples and their labels but also of *generalization* to unseen samples.

This chapter presents a very brief and introductory discussion of mathematical programming approach for solving binary (i.e. two class) data classification problem, and thereby take the readers to the arena of *support vector machines* (SVMs). The support vector machine algorithm has been developed by Vapnik [161] and is based on *statistical learning* theory. The SVM algorithms are non-parametric or data driven techniques which use minimum assumptions on the internal dynamics of the models. Some of the other popular nonparametric and data driven approaches are *multi layer perceptron (MLP)* and *projection pursuit regression (PPR)*. The major advantage of the SVM approach over other approaches (e.g. MLP and PPR) is that it results in a convex programming problem. These convex programming problems are generally structured, and efficient algorithms are available in the mathematical programming literature for their solution. Further, because of convexity, we always obtain a global optimal solution.

16.2 Binary (Two-Class) Pattern Classification Problems

By a pattern (data) we shall mean an element of $\mathbf{R}^n$. Let there be m patterns having class label +1 and k patterns having class label −1. Let A (respectively B) be the collection of all those patterns having class label +1 (respectively −1). Then we can construct matrix A (respectively B) of order $m \times n$ (respectively $k \times n$) by taking the ith row of A (respectively of B) as the ith pattern of class label +1 (respectively class label −1). We shall be using the same notation A (respectively B) for the set A (respectively B) as well as for the matrix A (respectively B) and the context will specify if we are refereing to the set or the matrix.

The problem of pattern classification aims to determine a criterion for distinguishing between elements of sets A and B by obtaining a decision surface in $\mathbf{R}^n$ which *separates* sets A and B. The decision surface which does the separation could be in the form of a hyperplane or a surface described by a nonlinear function.

Definition 16.2.1 ***(Linearly Separable Sets).*** *Two sets A and B of $\mathbf{R}^n$ are said to be linearly separable if their exists a hyperplane $w^T x = b, w \in \mathbf{R}^n, b \in \mathbf{R}$ such that*

$$Aw > eb \quad ; \quad Bw < eb\,. \tag{16.1}$$

In (16.1) A and B respectively are the matrices corresponding to the patterns of class +1 and class −1, and e is a vector of 'ones' of appropriate dimension.

If A and B are not linearly separable, then they are called *non-linearly separable*. The below given figure (Fig. 16.1) illustrates linearly and non-linearly separable datasets in $\mathbf{R}^2$.

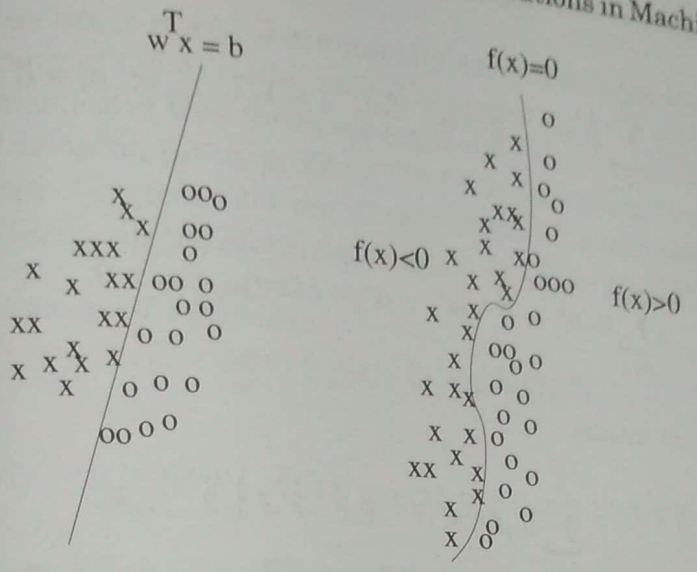


Fig. 16.1.

Lemma 16.2.1. *Let A and B be two finite sets in $\mathbf{R}^n$. Then A and B are linearly separable if and only if their convex hulls are disjoint.*

Proof. As A and B are finite sets in $\mathbf{R}^n$, their convex hulls are closed, bounded and convex. The proof then follows by employing the standard strict separation theorem (e.g. Mangasarian[110]).

Example 16.2.1 *Let $A = \{(-1,0),\ (0,1),\ (1,0)\}$ and $B = \{(0,0),\ (1,1),\ (0,2)\}$. Are A and B linearly separable?*

Solution As the convex hulls of A and B are not disjoint, the problem is NOT linearly separable.

Though Lemma 16.2.1 is useful in deciding if the given problem is linearly separable, it is not of much practical use, because in general, finding convex hulls is not an easy task. In the following we make use of linear programming to decide if the problem at hand is linearly separable. We first note that by a suitable scaling, the set of inequalities (16.1) can be rewritten as

$$Aw \geq eb + e$$

and,

$$Bw \leq eb - e. \tag{16.2}$$

This is because $Aw > eb$ means

$$\sum_{j=1}^{n} a_{ij} w_j > b \quad \text{for} \quad i = 1, 2, \ldots, m,$$

i.e.

$$\sum_{j=1}^{n} a_{ij}w_j \geq b + \alpha_i \quad \text{for} \quad i = 1, 2, \ldots, m \ , \alpha_i > 0$$

i.e.

$$\sum_{j=1}^{n} a_{ij}w_j \geq b + \alpha^*, \quad \alpha^* = \underset{i}{Min}(\alpha_i) \ , \alpha_i > 0. \tag{16.3}$$

Similarly $Bw < eb$ means

$$\sum_{j=1}^{n} b_{rj}w_j < b \quad \text{for} \quad r = 1, 2, \ldots, k$$

i.e.

$$\sum_{j=1}^{n} b_{rj}w_j \leq b - \beta_r \quad \text{for} \quad r = 1, 2, \ldots, k, \ \beta_r > 0$$

i.e.

$$\sum_{j=1}^{n} b_{rj}w_j \leq b - \beta^*, \ \beta^* = \underset{r}{Max}(\beta_r), \ \beta_r > 0 \tag{16.4}$$

If we now take $\overline{\alpha} = Min(\alpha^*, \beta^*)$, then (16.3) and (16.4) become

$$\begin{aligned} Aw &\geq eb + e \\ Bw &\leq eb - e \end{aligned} \tag{16.5}$$

where $w = w \ /\overline{\alpha}$ and $b = b \ /\overline{\alpha}$ are still denoted by w and b respectively.

As inequalities in (16.4) are '$\geq$' and '$\leq$' type, rather than '$>$' and '$<$' type, it is very natural to apply linear programming for the system (16.5).

Error Minimizing Linear Programming Problem

For a real number a, let $a_+ = \max(a, 0)$. Then for $x \in \mathbf{R}^n$ we write $x_+ = ((x_+)_1, \ldots, (x_+)_n)^T \in \mathbf{R}^n$ where $(x_+)_i = \max(x_i, 0), \ (i = 1, 2, \ldots, n)$. Also let $\|x\|_1$ denote the L_1 norm of the vector $x \in \mathbf{R}^n$, i.e. $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. We now consider the following optimization problem

$$\underset{(w,b)}{Min} \quad \frac{1}{m} \| \ (-Aw + eb + e)_+ \ \|_1 + \frac{1}{k} \| \ (Bw - eb + e)_+ \ \|_1 \ . \tag{16.6}$$

In (16.6) we note that if A and B are linearly separable, then the error is zero and hence the optimal value of (16.6) is also zero. The converse of the statement is also true. The above arguments give that the linear separability of the two sets A and B can be verified by solving the optimization problem (16.6). If the optimal value of the problem in (16.6) is zero then the given sets are linearly separable otherwise not. The below given Lemma 16.2.2 helps in transforming problem (16.6) in the LPP format

Lemma 16.2.2. *Consider the problems*

$$(I) \quad \underset{x \in S}{Min} \; \| g(x)_+ \|_1 + \| h(x)_+ \|_1$$

and

$$(II) \quad \underset{x \in S}{Min} \; \{e^T y + e^T z : y \geq g(x), y \geq 0, z \geq h(x), z \geq 0\}$$

where $S \subset \mathbf{R}^n$, $g : S \to \mathbf{R}^m$, $h : S \to \mathbf{R}^k$, $y \in \mathbf{R}^m$ *and* $z \in \mathbf{R}^k$. *Then both problems (I) and (II) have identical solution sets.*

Proof. We take $y = (g(x))_+$, i.e. $y = \max(g(x), 0)$. This gives $y \geq g(x)$, $y \geq 0$. Similarly, taking $z = (h(x))_+$ gives $z \geq h(x)$, $z \geq 0$. Also $\| g(x)_+ \|_1 = e^T y$ and $\| h(x)_+ \|_1 = e^T z$. This proves the Lemma. □

Now applying the above Lemma to the optimization problem (16.6), we get the following equivalent linear programming problem, called the *error minimizing linear programming problem*, as

$$\begin{aligned} \underset{w,b,y,z}{\text{Min}} \quad & \frac{e^T y}{m} + \frac{e^T z}{k} \\ \text{subject to} \quad & \\ & Aw - eb + y \geq e \\ & -Bw + eb + z \geq e \\ & y, z \geq 0, \\ & w \text{ and } b \text{ are unrestricted.} \end{aligned} \tag{16.7}$$

As problem (16.7) is a LPP, it can be solved efficiently by using the simplex algorithm and therefore can be used to check if the sets A and B are linearly separable or equivalently if the given pattern/data classification problem is linearly separable.

Some Useful deductions from the Error Minimizing LPP

Some useful deductions from problem (16.7) are

(i) Sets A and B are linearly separable if and only if the optimal value of the error minimizing LPP (16.7) is zero. Then the hyperplane $w^T x = b$ is the linear separator and the misclassification error is zero.

(ii) If A and B are linearly inseparable (i.e. not linearly separable), then the optimal value of error minimizing LPP (16.7) is not zero. In this case the sets A and B can not be separated by a hyperplane and the separation can be done by a nonlinear separator only. Further the optimal solution of (16.7) will provide a hyperplane $w^T x = b$ that does not separate the sets A and B exactly but for which the average error of misclassification (i.e. the objective function of problem (16.7)) is least.

(iii) The solution $(w = 0, b, y, z)$ is an optimal solution of (16.7) if and only if $\frac{e^T A}{m} = \frac{e^T B}{k}$, in which case it is never unique in $w = 0$. Thus there always exists a solution of the error minimizing LPP (16.7) with a nonzero w (Bennett and Mangasarian [17]).

Example 16.2.2 ***(AND Problem).*** *Write the error minimizing LPP for the AND problem and check if the given problem is linearly separable.*

Solution For the AND logic, we have $A = (1 \quad 1)$ and

$$\mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Hence the error minimizing LPP for the patterns A and B is

$$\begin{aligned} \textit{Min} \quad & y_1 + \frac{1}{3}(z_1 + z_2 + z_3) \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} w_1 + w_2 - b + y_1 &\geq 1 \\ b + z_1 &\geq 1 \\ -w_2 + b + z_2 &\geq 1 \\ -w_1 + b + z_3 &\geq 1 \\ y_1, z_1, z_2, z_3 &\geq 0, \\ w_1, w_2, b &\quad \text{unrestricted in sign.} \end{aligned}$$

An optimal solution of the above LPP is $\overline{w_1} = 2$, $\overline{w_2} = 2$, $\overline{b} = 3$, $\overline{y_1} = 0$, $\overline{z_1} = 0$, $\overline{z_2} = 0$, and $\overline{z_3} = 0$. As the optimal value of the error minimizing LPP is zero, the given problem is linearly separable and the linear classifier is given by $\overline{w_1}x_1 + \overline{w_2}x_2 = b$, i.e. $x_1 + x_2 = 1.5$. This may also be understood geometrically as shown in Fig 16.2.

The convex hull of the set A is the singleton $\{(1,1)\}$ and that of the set B is the solid triangle PQR. As these convex hulls are disjoint, the AND problem is linearly separable.

Example 16.2.3 ***(XOR Problem).*** *Write the error minimizing LPP for the Exclusive-OR (XOR) problem and check if the given problem is linearly separable.*

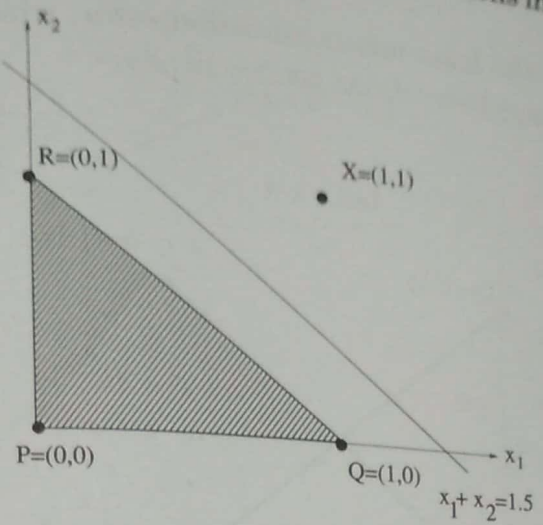


Fig. 16.2.

Solution For the XOR logic, we have

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

Hence the error minimizing LPP is

$$\begin{aligned} \textit{Min} \quad & \frac{1}{2}(y_1 + y_2) + \frac{1}{2}(z_1 + z_2) \\ \text{subject to} \quad & \\ & w_1 - b + y_1 \geq 1 \\ & w_2 - b + y_2 \geq 1 \\ & b + z_1 \geq 1 \\ & -w_1 - w_2 + b + z_2 \geq 1 \\ & y_1, y_2, z_1, z_2 \geq 0, \\ & w_1, w_2, b \quad \text{unrestricted in sign.} \end{aligned}$$

Here $\frac{e^T A}{2} = \frac{e^T B}{2}$ and $(w = 0\ ,\ b = 0\ , y = e\ , z = e)$ is an optimal solution of the above LPP with the optimal value as 2. Therefore the XOR problem is NOT linearly separable. But we also know that there certainly exists optimal solution of the above LPP with nontrivial $\overline{w}$. In fact $(\overline{w_1} = 2, \overline{w_2} = 2, \overline{b} = 1, \overline{y_1} = 0, \overline{y_2} = 0, \overline{z_1} = 0,$ and $\overline{z_2} = 4.)$ is an alternate optimal solution which gives the hyperplane (line) $x_1 + x_2 = 0.5$. As the problem is not linearly separable, this line will not exactly separate the patterns of class

A and B, but it will give the least misclassification error, which in this case equals 2. Figure 16.3 illustrates these observations geometrically.

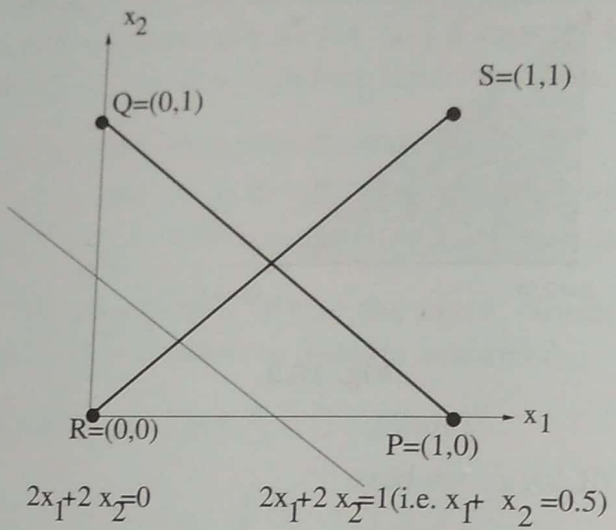


Fig. 16.3.

The line segments PQ and RS are the convex hulls of the sets A and B respectively. As these convex hulls are not disjoint, the problem is not linearly separable. Further, the only point not correctly classified by the classifier $2x_1 + 2x_2 = 1$, i.e. the line $x_1 + x_2 = 0.5$, is the point (1,1). The misclassification error for the point (1,1) is $|w^T x - (b-1)|$ where $w_1 = w_2 = 2$, $b = 1$ and $x = (1,1)^T$, giving the error equal to 4. But as per our construction of the error minimizing LPP (16.7) the least error of misclassification is taken the average error of misclassification, i.e, $\frac{1}{2}(0+0) + \frac{1}{2}(0+4) = 2$.

16.3 Optimal Separation For Linearly Separable Data Sets

Let the sets $A, B \subset \mathbf{R}^n$ be linearly separable. Then there certainly exists a separating hyperplane $w^T x = b$ which strictly separates sets A and B. In fact for a linearly separable classification problem, there will be infinitely many separating hyperplanes.

So how should we choose the separating hyperplane? Can we talk of an optimal separating hyperplane? These are the questions which we wish to answer in this section. For this we first introduce the concept of *dead zone* and *margin*.

Definition 16.3.1. (Canonical Hyperplane). *Let $A, B \subset \mathbf{R}^n$ be linearly separable. Then a separating hyperplane $w^T x = b$ is called a canonical separating hyperplane if it satisfies $Aw \geq eb + e$ and, $Bw \leq eb - e$.*

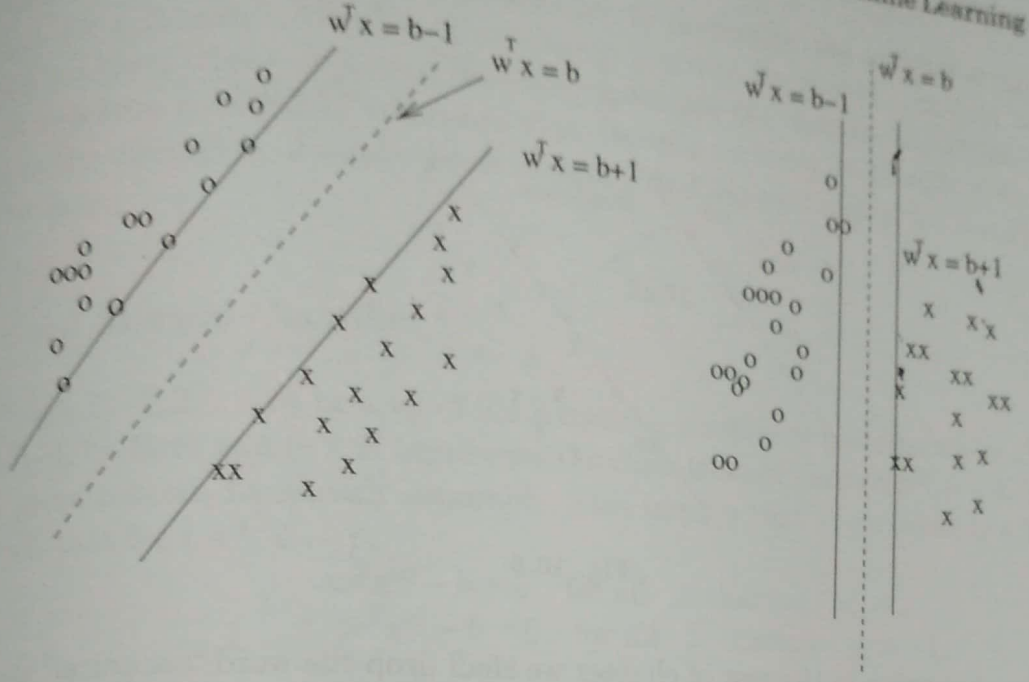


Fig. 16.4.

Definition 16.3.2. (Dead Zone). *Let $A, B \subset \mathbf{R}^n$ be linearly separable and $w^T x = b$ be a canonical separating hyperplane. Then there definitely exists a region $\{x : (b-1) < w^T x < (b+1)\} \subset \mathbf{R}^n$ surrounding the separating hyperplane $w^T x = b$ which is void of points from the sets A and B. This region is called the dead zone for the separating hyperplane $w^T x = b$.*

Definition 16.3.3. (Margin). *Let $A, B \subset \mathbf{R}^n$ be linearly separable and $w^T x = b$ be a canonical separating hyperplane. Let $w^T x = (b-1)$ and $w^T x = (b+1)$ be the bounding hyperplanes which define the dead zone. Then the distance between these two bounding planes, namely $w^T x = (b-1)$ and $w^T x = (b+1)$, is called the margin for the separating hyperplane $w^T x = b$.*

It is easy to compute the margin for a given canonical separating hyperplane $w^T x = b$. We simply compute the perpendicular distance between a point on the hyperplane $w^T x = (b-1)$ and the hyperplane $w^T x = b$, which equals $\frac{1}{\|w\|}$. Therefore for the separating hyperplane $w^T x = b$, the margin is $\frac{2}{\|w\|}$, where $\|w\|^2 = (w_1^2 + \ldots, + w_n^2)$. Figure 16.5 gives a pictorial illustration of the dead zone and margin for the separating hyperplane $w^T x = b$.

Definition 16.3.1 (Optimal Separating Hyperplane). *Let sets $A, B \subset \mathbf{R}^n$ be linearly separable and $w^T x = b$ be a canonical separating hyperplane satisfying (16.2). Then the separating hyperplane $w^T x = b$ is called optimal separating hyperplane if its margin is the maximum amongst all canonical separating hyperplanes.*

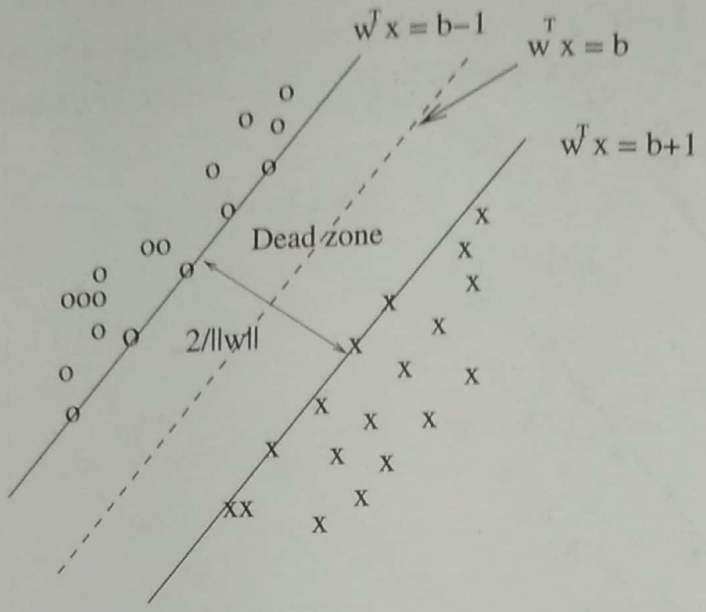


Fig. 16.5.

For our discussion in the rest of chapter we shall drop the word 'canonical' and refer to a separating hyperplane as a canonical separating hyperplane only.

The obvious question now is with regard to the definition of optimality in terms of margin. Why should we be interested in that separating hyperplane for which the margin is a maximum ? There is a very sound theoretical justification for the choice, that involves some deeper concepts (e.g. *V.C. dimension* and *structural risk minimization*) from the statistical theory of machine learning. We refer to the interested reader to some other materia, mentioned in Section 16.7. In classification, our aim is not only to classify the available training data in accordance with the class label, but also to have good generalization on unseen, or test data. Learning assumes that training and test data are drawn from the same distribution. However, with a finite number of training data, there is bound to be some generalization error. The larger the dead zone, the less likely it is that a test data point will lie on the wrong side of the separating surface or decision boundary. Therefore, we would like to maximize the dead zone which implies that we should have the maximum margin.

In view of the above, in a linearly separable classification problem, our aim is not only to find a separating hyperplane (i.e. a hyperplane for which the classification error is zero), but also for which the margin is a maximum. We have already seen that for a linearly separable problem, the error minimizing LPP (16.6) will give a separating hyperplane. But does it always give the optimal separating hyperplane? The answer is in the negative. For the AND problem, the classifier given by the error minimizing LPP, namely $x_1 + x_2 = 1.5$, is certainly optimal. But if we take $A = \{(10,1)\}$, $B = \{(8,3)\}$ in $\mathbf{R}^2$ and solve the error minimizing LPP, we get the classifier (separating line) as $0.181x_1 - 0.818x_2 = 0$. This classifier can NOT be optimal because geometrically, for maximizing the margin, the classifier has to be the perpendicular bisector of the points (10,1) and (8,3), i.e. $x_1 - x_2 = 7$.

The above discussion suggests, that in order to determine the optimal separating hyperplane for a linearly separable classification problem we should construct an optimization problem whose objective function represents the margin. Additionally, we should incorporate constraints to ensure that there is no error in classification. This leads us to the discussion of *hard margin classifier* and *support vectors* for linearly separable datasets.

16.4 Hard Margin Classifier

Let $\{(x^{(i)}, y_i),\ i = 1, 2, \ldots, p\}$ be a set of finite training samples or patterns where $x^{(i)} \in \mathbf{R}^n$ and $y_i \in \{-1, 1\}$. Here $y_i \in \{-1, 1\}$ represents the class label of pattern $x^{(i)}, (i = 1, 2, \ldots p)$. Let the given pattern be linearly separable. This implies that there exists $w \in \mathbf{R}^n$ and $b \in \mathbf{R}$ such that for $i = 1, 2, \ldots p$,

$$w^T x^{(i)} - b > 0 \quad \text{for all} \quad i \quad \text{having} \quad y_i = 1$$

and

$$w^T x^{(i)} - b < 0 \quad \text{for all} \quad i \quad \text{having} \quad y_i = -1.$$

By suitable scaling we can write the above as

$$w^T x^{(i)} - b \geq 1 \qquad \text{for all} \quad i \quad \text{having} \quad y_i = 1$$

and

$$w^T x^{(i)} - b \leq -1 \qquad \text{for all} \quad i \quad \text{having} \quad y_i = -1.$$

This can be written in a more compact form as

$$y_i(w^T x^{(i)} - b) \geq 1 \quad \text{for all} \quad i = 1, 2, \ldots p. \tag{16.8}$$

Note that inequalities (16.8) are the same as the constraints of the error minimizing LPP (16.7), except that the error variables are taken as zero, because the problem is linearly separable and $w^T x = b$ is a separating hyperplane. We have of course changed our notation a bit for the sake of convenience. We have not formed matrices A and B separately, because we have taken all the patterns together in constraints (16.8), and the sign of the ith constraint gets associated with '$\geq$' or '$\leq$', depending upon whether $y_1 = 1$ or -1.

Amongst all separating hyperplanes $w^T x = b$, we have to choose the one for which the margin $\frac{2}{\|w\|}$ is a maximum, or equivalently $\frac{\|w\|}{2}$ is a minimum. However, minimizing $\frac{\|w\|}{2}$ is equivalent to minimizing $\frac{\|w\|^2}{2}$ and therefore the optimization problem for determining the maximum margin classifier is given by

$$\begin{aligned} \underset{(w,b)}{\text{Min}} \quad & \frac{1}{2} w^T w \\ \text{subject to} \quad & y_i(w^T x^{(i)} - b) \geq 1 \qquad (i = 1, 2, \ldots, p). \end{aligned} \tag{16.9}$$

Problem (16.9) is a standard convex quadratic programming problem (QPP) which can be solved by a suitable QPP algorithm. Once an optimal solution $(\overline{w}, \overline{b})$ of (16.9) has been obtained, the maximum margin classifier $\overline{w}^T x = \overline{b}$ is known. This classifier is also known as the *hard margin classifier*. There is one practical difficulty with QPP (16.9). It has as many constrains as the number of patterns and hence may be extremely difficult to solve for large datasets. In the machine learning literature, special *chunking type algorithms and decomposition schemes* have been developed so that not all constraints are included at one time. In mathematical programming, the standard approach to handle a large number of constraints is to examine if the dual problem can be solved efficiently. We attempt to use the same strategy for (16.9) and write the dual. This requires the Karush-Kuhn-Tucker (KKT) conditions for problem (16.9).

KKT conditions for QPP (16.9)

The Lagrangian for problem (16.9) is given by

$$L(w, b, \alpha) = \frac{1}{2} w^T w + \sum_{i=1}^{p} \alpha_i (1 - y_i (w^T x^{(i)} - b)), \tag{16.10}$$

where $w \in \mathbf{R}^n$, $b \in \mathbf{R}$ and $\alpha \in \mathbf{R}^p$ are the vectors of Lagrange multipliers. Then the KKT conditions are

$$\nabla_w L = w - \sum_{i=1}^{p} \alpha_i y_i x^{(i)} = 0 \tag{16.11}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{p} \alpha_i y_i = 0 \tag{16.12}$$

$$1 - y_i (w^T x^{(i)} - b) \leq 0 \quad (i = 1, 2, \ldots, p) \tag{16.13}$$

$$\alpha_i [1 - y_i (w^T x^{(i)} - b)] = 0 \quad (i = 1, 2, \ldots, p) \tag{16.14}$$

$$\alpha_i \geq 0 \quad (i = 1, 2, \ldots, p)\,. \tag{16.15}$$

Let $(\overline{w}, \overline{b}, \overline{\alpha})$ be a solution to the above KKT system. Then, by the KKT theorem, $(\overline{w}, \overline{b})$ is optimal to the (convex) quadratic programming problem (16.9). Using the KKT conditions (16.12)-(16.15) we have

$$\overline{w} = \sum_{i=1}^{p} \overline{\alpha_i} y_i x^{(i)} \tag{16.16}$$

and $y_i (\overline{w}^T x^{(i)} - \overline{b}) = 1$ for those patterns $x^{(i)}$ having $\overline{\alpha_i} > 0$.

Therefore, the vector $\overline{w}$ is known as soon as the $\overline{\alpha_i}$'s are known. If some $\overline{\alpha_i} > 0$, then the corresponding pattern $x^{(i)}$ lies on one of the bounding planes $\overline{w}^T x = (\overline{b} - 1)$ and $\overline{w}^T x = (\overline{b} + 1)$. These patterns, i.e. the ones for which $\overline{\alpha_i} > 0$, are called the *support vectors*. Geometrically these are patterns $x^{(i)}$ which lie on one of the bounding planes $\overline{w}^T x = (\overline{b} - 1)$, or $\overline{w}^T x = (\overline{b} + 1)$. From Fig 16.2, we observe that patterns (1, 0), (0, 1) and (1, 1) are support vectors for the hard margin classifier of the AND problem. This is because (1, 0) and (0, 1) lie on the line $2x_1 + 2x_2 = 0$ and (1,1) lies on line $2x_1 + 2x_2 = 2$. The two lines are the two bounding planes. The lagrange multipliers corresponding to the patterns (0, 0) (1, 0) and (0, 1) (which are support vectors) are bound to be positive.

Let S denote the index set of all support vectors, i.e.

$$\begin{aligned} S = & \{i : x^{(i)} \text{is a support vector}, 1 \le i \le p\} \\ = & \{i : \overline{\alpha_i} > 0, 1 \le i \le p\} \end{aligned} \tag{16.17}$$

Then $\overline{w} = \sum_{i \in S} \overline{\alpha}_i y_i x^{(i)}$ because for $i \notin S$, $\overline{\alpha_i} = 0$. If we now pick a specific support vector, say, $x^{(k)}$ then $\alpha^{(k)} > 0$ implies $y_k(\overline{w}^T x^{(k)} - \overline{b}) = 1$. This gives $y_k^2(\overline{w}^T x^{(k)} - \overline{b}) = y_k$, i.e. $\overline{w}^T x^{(k)} - \overline{b} = y_k$ because $y_k^2 = 1$. This implies that $\overline{b} = -y_k + \overline{w}^T x^{(k)}$. Since there could be many support vectors, in practice, we compute the values of $\overline{b}$ corresponding to each support vector, and then take the average of all these values.

For any unseen or test point $x^{(p+1)}$, we compute $\overline{w}^T x^{(p+1)}$. If $\overline{w}^T x^{(p+1)} - \overline{b} > 0$ then $x^{(p+1)}$ is labeled as +1, while if it is less than zero then $x^{(p+1)}$ is labeled as -1. Thus once the classifier $\overline{w}^T x = \overline{b}$ is known, the whole process of assigning label to a future pattern works like a machine. Since only the support vectors are used in the determination of the classifier $\overline{w}^T x = \overline{b}$, the learning machine built around this approach is called a *support vector machine (SVM)*.

The SVM algorithm as described above clearly needs the knowledge of the KKT multipliers $\overline{\alpha}_i, (i = 1, 2, \ldots, p)$. Once the $\overline{\alpha_i}$'s are known, we know the support vectors and hence $\overline{w}$ and $\overline{b}$. How can be determine these multipliers? Recalling our discussion on nonlinear programming duality (Chapter 8), we note that KKT multipliers $\overline{\alpha_i}$ are nothing but the dual variables. Therefore, we write the Wolfe dual of problem (16.9).

Wolfe dual of Problem (16.9)

The dual of problem (16.9) is given by

$$\begin{aligned} \text{Max} \quad & L(w,b,\alpha) \\ \text{subject to} \quad & \\ & \nabla_w L(w,b,\alpha) = 0 \\ & \frac{\partial L}{\partial b} = 0 \\ & \alpha \geq 0, \end{aligned} \tag{16.18}$$

where $L(w,b,\alpha)$ is defined in (16.10). From (16.11) and (16.12), we have

$$w = \sum_{i=1}^{p} \alpha_i y_i x^{(i)} \tag{16.19}$$

and $\sum_{i=1}^{p} \alpha_i y_i = 0$. Therefore using (16.19) we get

$$\begin{aligned} L &= \frac{1}{2} w^T w + \sum_{i=1}^{p} \alpha_i (1 - y_i (w^T x^{(i)} - b)) \\ &= \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j {x^{(i)}}^T x^{(j)} - \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j {x^{(i)}}^T x^{(j)} + b \sum_{i=1}^{p} \alpha_i y_i + \sum_{i=1}^{p} \alpha_i \\ &= \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j {x^{(i)}}^T x^{(j)} \end{aligned} \tag{16.20}$$

because $\sum_{i=1}^{p} \alpha_i y_i = 0$.

This gives the dual problem (16.18) as

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j {x^{(i)}}^T x^{(j)} \\ \text{subject to} \quad & \\ & \sum_{i=1}^{p} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \ (i = 1, 2, \ldots, p). \end{aligned} \tag{16.21}$$

In problem (16.21) we note that apart from the non-negativity constraint $\alpha \geq 0$, there is only one other constraint. The objective function of (16.21) is concave. Therefore (16.21) is probably easier to solve than the (primal) problem (16.9). Once the optimal solution $\overline{\alpha}$ of (16.21) is known, the values of $\overline{w}$ and $\overline{b}$ can be computed and hence the separating hyperplane $\overline{w}^T x = \overline{b}$ can be determined.

Example 16.4.1. Let $A = \{(1,8), (4,5), (4,4), (1,1)\}$ and $B = \{(8,3)\}$ be two sets in $\mathbf{R}^2$. Show that (i) A and B are linearly separable. (ii) Obtain the hard margin classifier by solving the primal problem. (iii) Obtain the hard margin classifier by solving the dual problem.

Solution. The linear separability can be checked either by finding the convex hull (geometrically) or by solving the error minimizing LPP. It is left to the readers to verify that sets A and B are linearly separable. We now determine the hard margin classifier by solving the primal problem as well as the dual problem. The primal problem is

$$\begin{aligned} \textit{Min} \quad & \frac{1}{2} w_1^2 + \frac{1}{2} w_2^2 \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} w_1 + 8w_2 - b &\geq 1 \\ 4w_1 + 5w_2 - b &\geq 1 \\ 4w_1 + 4w_2 - b &\geq 1 \\ w_1 + w_2 - b &\geq 1 \\ -8w_1 - 3w_2 + b &\geq 1 \\ w_1, w_2, b &\quad \text{are unrestricted.} \end{aligned} \tag{16.18}$$

We can solve the above QPP by Wolfe's method to get $\overline{w_1} = -0.47$, $\overline{w_2} = 0.12$, $\overline{b} = -2.41$. Therefore the hard margin classifier (i.e. the separating hyperplane with maximum margin) is $-0.47x_1 + 0.12x_2 = -2.41$, i.e. $0.47x_1 - 0.12x_2 = 2.41$.

The dual formulation (16.21) for the above problem is

$$\begin{aligned} \textit{Max} \quad & -\frac{65}{2}\alpha_1^2 - \frac{41}{2}\alpha_2^2 - 16\,\alpha_3^2 - \alpha_4^2 - \frac{73}{2}\alpha_5^2 - 44\alpha_1\alpha_2 \\ & -36\alpha_1\alpha_3 - 9\alpha_1\alpha_4 + 32\alpha_1\alpha_5 - 36\alpha_2\alpha_3 - 9\alpha_2\alpha_4 + 47\alpha_2\alpha_5 - 8\alpha_3\alpha_4 \\ & +44\alpha_3\alpha_5 + 11\alpha_4\alpha_5 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 \end{aligned} \tag{16.19}$$

subject to

$$\begin{aligned} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5 &= 0 \\ \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 &\geq 0 \end{aligned} \tag{16.20}$$

whose optimal solution is $\overline{\alpha_1} = 0$, $\overline{\alpha_2} = 0$, $\overline{\alpha_3} = 0.12$, $\overline{\alpha_4} = 0$, and $\overline{\alpha_5} = 0.12$. Therefore patterns $x^{(3)} = (4,4)$ and $x^{(5)} = (8,3)$ are support vectors, and

(16.21)

$$\begin{aligned}\overline{w} &= \overline{\alpha_3} y_3 x^{(3)} + \overline{\alpha_5} y_5 x^{(5)} \\ &= (0.12)(1)(4,4)^T + 0.12(-1)(8,3)^T \\ &= (-0.48, 0.12)^T .\end{aligned}$$

Furthermore, $\overline{b} = 0.5(\overline{b_3} + \overline{b_5})$ where $\overline{b_3}$ and $\overline{b_5}$ are given by

$$\begin{aligned}\overline{b_3} &= -y_3 + \overline{w}^T x^{(3)} = -1 + (-0.48, 0.12)(4,4)^T \\ \overline{b_5} &= y_5 - \overline{w}^T x^{(5)} = -(-1) + (-0.48, 0.12)(8,3)^T\end{aligned}$$

This gives $\overline{b} = -2.46$. Therefore the hard margin classifier is $0.48x_1 - 0.12x_2 = 2.46$, which is the same as the one given by the primal problem, except for some differences due to numerical computations.

By solving the error minimizing LPP, we not only check that the given problem is linearly separable but also determine the separating hyperplane. For our present example, the error minimizing LPP gives the separating hyperplane as $4x_1 - 4x_2 = 1$ which is different from the one we have obtained by solving the hard margin classifier problem. This example again illustrate that the error minimizing LPP need not yield the maximum margin classifier.

16.5 Soft Margin Classifier

Let $\{(x^{(i)}, y_i),\ i = 1,2,\ldots,p\}$ be a finite training sample of patterns where $x^{(i)} \in \mathbf{R}^n$ and $y_i \in \{-1, 1\}$. We now consider the case when these patterns are not linearly separable. In this situation the error minimizing LPP (16.7) will have a nonzero objective function value, i.e. in the optimal solution of (16.7) not all error variables will be zero. Let these error variables be denoted by ξ_i $(i = 1,2,\ldots,p)$. (Here again there is a slight change of notation from problem (16.7), in the sense that all error variables are denoted by ξ_i rather than having two different set of error variables for the two different classes). Now our aim is to find a classifier $w^T x = b$ for which the total error $\sum_{i=1}^{p} \xi_i$ is least and the margin $\frac{2}{\|w\|}$ is maximum. This is a typical scenario for multiobjective optimization: we have to find a classifier $w^T x = b$ for which both $\sum_{i=1}^{p} \xi_i$ and $\frac{w^T w}{2}$ are minimum. Since this is not possible in general, we consider the weighted sum $\frac{w^T w}{2} + C\sum_{i=1}^{p} \xi_i$ and try to find a trade off between the misclassification error and the margin, by choosing the parameter $C > 0$ appropriately. This leads us to the following optimization problem

$$\underset{(w,b,\xi)}{Min} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{p}\xi_i$$

subject to

$$y_i(w^T x^{(i)} - b) + \xi_i \geq 1 \quad (i = 1,2,\ldots,p)$$
$$\xi_i \geq 0, \quad (i = 1,2,\ldots,p). \tag{16.23}$$

Here, the constraints are same as in the error minimizing LPP (16.7), and our preference for the margin/misclassfiaction error will be indicated by our choice of C. If we choose $C > 0$ to be large, then this will imply that we are placing a greater emphasis on the misclassification error. In this case, our classifier $w^T x = b$ will possibly classify most of the training samples correctly, but will have a smaller margin and therefore may not generalize as well. There can not be any specific rule for the choice of C, and it will depend upon how much tradeoff we wish to have between the two objectives, namely, the margin and the misclassification error.

The quadratic programming problem (16.23) is referred as the *soft margin classifier*, because in this case, the margin is no longer 'hard', it depends on the error variable ξ_i.

On the lines of the hard margin classifier, we define the Lagrangian as

$$L(w,b,\xi,\alpha) = \frac{1}{2}w^T w + C(\sum_{i=1}^{p}\xi_i) + \sum_{i=1}^{p}\alpha_i(1 - \xi_i - y_i(w^T x^{(i)} - b)) - \sum_{i=1}^{p}\beta_i\xi_i, \tag{16.24}$$

and obtain the following KKT conditions

$$\nabla_w L = w - \sum_{i=1}^{p}\alpha_i y_i x^{(i)} = 0 \tag{16.25}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{p}\alpha_i y_i = 0 \tag{16.26}$$

$$\frac{\partial L}{\partial \xi} = C - \alpha_i - \beta_i = 0\ ,(i = 1,2,\ldots,p) \tag{16.27}$$

$$\xi_i - y_i(w^T x^{(i)} - b) \geq 1\ ,(i = 1,2,\ldots,p) \tag{16.28}$$

$$\alpha_i[1 - \xi_i - y_i(w^T x^{(i)} - b)] = 0\ ,(i = 1,2,\ldots,p) \tag{16.29}$$

$$\beta_i\xi_i = 0 \quad ,(i = 1,2,\ldots,p). \tag{16.30}$$

$$\xi_i,\ \alpha_i,\ \beta_i \geq 0\ ,(i = 1,2,\ldots,p) \tag{16.31}$$

$$\alpha, \beta, \xi \geq 0, w, b \text{ unrestricted sign.} \tag{16.32}$$

The above KKT conditions reveal some interesting points. Let $(\overline{w}, \overline{b}, \overline{\alpha}, \overline{\beta}, \overline{\xi})$ be a solution of the system (16.25)-(16.31). Now (16.27)-(16.30) give $(C - \overline{\alpha_i})\overline{\xi_i} = 0$. Therefore for $0 \leq \overline{\alpha_i} < C$, we get $\overline{\xi_i} = 0$, i.e. there is no error in the classification of the patterns $x^{(i)}$.

Specifically we consider the following three cases.
Case 1 Let $0 < \overline{\alpha_i} < C$. In this case $\overline{\xi_i} = 0$ and so the pattern $x^{(i)}$ is correctly classified. From (16.29), we have $y_i(\overline{w}^T x^{(i)} - \overline{b}) = 1$, i.e. the pattern $x^{(i)}$ lies on the bounding planes $w^T x^{(i)} = b + 1$ or $w^T x^{(i)} = b - 1$. Such patterns $x^{(i)}$ are called *free support vectors.*
Case 2 Let $\overline{\alpha_i} = 0$. Then (16.27) gives $\overline{\beta_i} = C > 0$ and hence (16.30) gives $\overline{\xi_i} = 0$, i.e. the pattern $x^{(i)}$ is correctly classified. Also from (16.28) we get $y_i(\overline{w}^T x^{(i)} - b) \geq 1$.
Case 3 Let $\overline{\alpha_i} = C$. Then (16.27) gives $\overline{\beta_i} = 0$. Also from (16.29), $\alpha_i(1 - \xi_i - y_i(w^T x^{(i)} - b)) = 0$, i.e. $(1 - \xi_i - y_i(w^T x^{(i)} - b)) = 0$ as $\alpha_i > 0$. But $\xi_i \geq 0$ and hence for $0 \leq \xi_i < 1$, we get $y_i(w^T x^{(i)} - b) = (1 - \xi_i) > 0$. This implies that though $x^{(i)}$ is correctly classified, it lies inside the dead zone. However for the case $\xi_i > 1$, we have $y_i(w^T x^{(i)} - b) = (1 - \xi_i) < 0$ and therefore the patterns $x^{(i)}$ is not correctly classified. The patterns $x^{(i)}$ corresponding to $\overline{\alpha_i} = C$ are called *bounded support vectors.*

Remark 16.5.1 *In the case of the hard margin classifier, all* $\overline{\xi_i} = 0$, *and therefore for the linearly separable case, the margin is free of any pattern* $x^{(i)}$. *In the soft margin case, some patterns will lie in the margin (dead zone).*

Example 16.5.1. Consider the patterns • (positive class) and ■ (negative class) as shown in Fig 16.6. Let $\overline{w}^T x = \overline{b}$ be a soft margin classifier obtained for a particular choice of C. Identify (i) free and bounded support vectors (ii)the misclassified points of positive and negative class (iii) the possible values of $\overline{\alpha_i}$ and $\overline{\xi_i}$ for each pattern $x^{(i)}$.

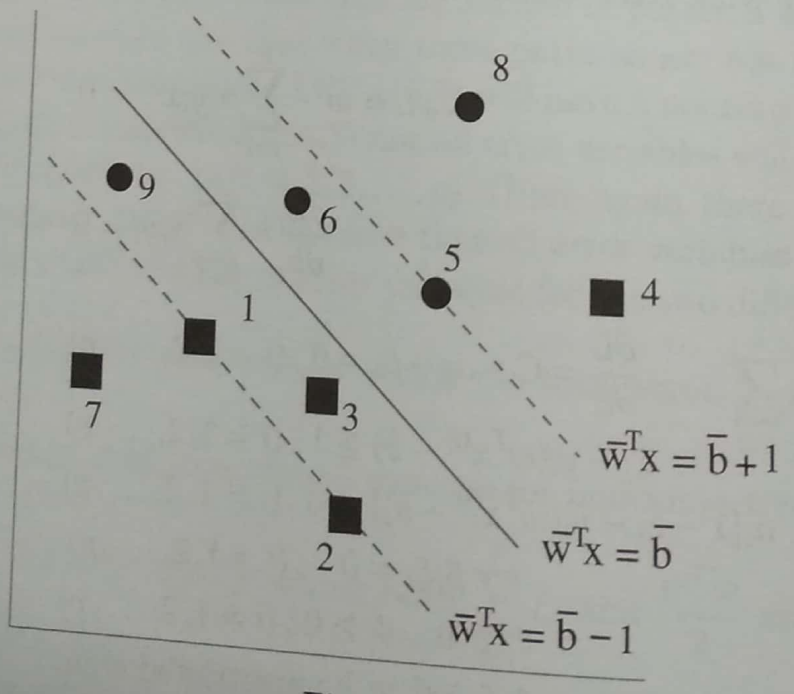


Fig. 16.6.

Solution.

Patterns $x^{(i)}$	Misclassification Error ξ_i	Multiplier λ_i	Nature of $x^{(i)}$
$x^{(1)}$	$\xi_1 = 0$	$0 < \alpha_1 < C$	free support vectors lying on the hyperplane $\bar{w}^T x = (\bar{b} - 1)$.
$x^{(2)}$	$\xi_2 = 0$	$0 < \alpha_2 < C$	free support vectors lying on the hyperplane $\bar{w}^T x = (\bar{b} - 1)$.
$x^{(5)}$	$\xi_5 = 0$	$0 < \alpha_5 < C$	free support vectors lying on the hyperplane $\bar{w}^T x = (\bar{b} + 1)$.
$x^{(6)}$	$0 < \xi_6 < 1$	$\alpha_6 = C$	Positive class point which is correctly classified. Also it is a bounded support vector
$x^{(3)}$	$0 < \xi_3 < 1$	$\alpha_3 = C$	Negative class point which is correctly classified. Also it is a bounded support vector.
$x^{(4)}$	$\xi_4 > 1$	$\alpha_4 = C$	Negative class point which is misclassified. Also it is a bounded support vector.
$x^{(7)}$	$\xi_7 = 0$	$\alpha_7 = 0$	Negative class point which is correctly classified.
$x^{(8)}$	$\xi_8 = 0$	$\alpha_8 = 0$	Positive class point which is correctly classified.
$x^{(9)}$	$\xi_9 > 1$	$\alpha_9 = C$	Positive class point which is misclassified. Also it is a bounded support vector.

Using the Lagrangian $L(w, b, \xi, \alpha)$ given in (16.24), we obtain the following dual of problem (16.23)

$$\text{Max} \quad -\frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}\alpha_i\alpha_j y_i y_j x^{(i)^T}x^{(j)} - b\sum_{i=1}^{p}\alpha_i y_i + \sum_{i=1}^{p}\alpha_i + \sum_{i=1}^{p}(C - \alpha_i - \beta_i)\xi_i$$

subject to

$$\sum_{i=1}^{p}\alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad (i = 1, 2, \ldots, p)$$

$$C - \alpha_i \geq 0 \quad (i = 1, 2, \ldots, p).$$

i.e.

$$\text{Max} \quad \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j x^{(i)^T} x^{(j)}$$

subject to

$$\sum_{i=1}^{p} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad (i = 1, 2, \ldots, p). \tag{16.33}$$

The dual problem (16.33) can be solved by the usual QPP techniques to get $\bar{\alpha}$. Once optimal $\bar{\alpha}$ is known we get $\bar{w} = \sum_{i=1}^{p} \bar{\alpha}_i y_i x^{(i)}$ and $\bar{b} = -y_j + (\bar{w})^T x^{(j)}$ for some j such that $0 < \bar{\alpha}_j < C$.

16.6 Nonlinear Support Vector Machines

The support vector machines discussed in Sections 16.4 and 16.5 are called linear SVMs because the discriminant function is linear (a hyperplane $w^T x = b$). When the patterns are not linearly separable, then depending upon the value of C, a soft margin classifier $w^T x = b$ is obtained, for which the error on the training set is the least. However, in many applications this 'least' error may be unacceptably high. In such a case, we need to find a suitable nonlinear discriminant function or nonlinear decision surface. This is achieved by the Kernel SVM approach which employs the so called *kernelization* or *kernel trick*.

To motivate the kernel approach, let us consider the classical XOR problem. Here the training patterns are $\{(0,0),(1,1),\ (0,1),(1,0)\}$ with $\{(0,0),(1,1)\}$ having class label $+1$ and $\{(0,1),(1,0)\}$ having class label -1. We have seen that this problem is not linearly separable. The question which we now pose is as follows: Is it possible to think of a mapping ϕ from the *input space* (the space in which the training samples lie, which is $\mathbf{R}^2$ in the case of XOR) to a suitable *feature space* (a higher dimensional space $\mathbf{R}^p$ $(p > 2)$) so that in the feature space the patterns $\{\phi(0,0), \phi(1,1),\ \phi(0,1), \phi(1,0)\}$ are linearly separable? If the answer to this question is 'yes' then we can apply the linear SVM approach of Sections 16.4 and 16.5 to the transformed patterns in the feature space. Let us try $\phi : \mathbf{R}^2 \to \mathbf{R}^3$ given by $\phi(x_1, x_2) = (x_1^2\ ,\ \sqrt{x_1 x_2}\ , x_2^2)$. This gives $\phi(0,\ 0) = (0,\ 0,\ 0)$, $\phi(1,\ 1) = (1,\ 1,\ 1)$, $\phi(1,\ 0) = (1,\ 0,\ 0)$, and $\phi(0,\ 1) = (0,\ 0,\ 1)$. It can easily be visualized in $\mathbf{R}^3$ that the convex hulls of the sets $\{(0,\ 0,\ 0),\ (1,\ 1,\ 1)\}$ and $\{(1,\ 0,\ 0),\ (0,\ 0,\ 1)\}$ are disjoint and so the sets are linearly separable.

But is the above approach always valid, i.e. will there always exist $\phi : \mathbf{R}^n \to \mathbf{R}^p$, $(p > n)$ so that the transformed patterns in the feature space $\mathbf{R}^p$ are linearly separable? In other words, is Fig. 16.7 always correct?

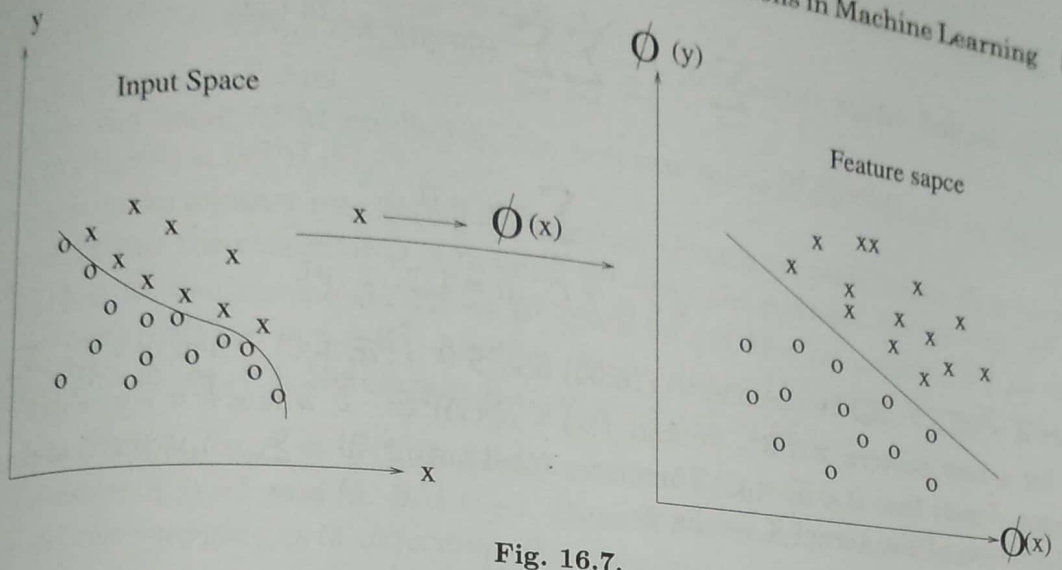


Fig. 16.7.

Let us suppose that such a ϕ exists (we shall justify its existence later). We can then obtain the optimal hyperplane for the transformed training set $\{(z^{(i)}, y_i) : z^{(i)} = \phi(x^{(i)}),\ i = 1, 2, \ldots, p\}$ by solving problem (16.33) with $x^{(i)}$ replaced by $z^{(i)}$, i.e.

$$\text{Max} \quad \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j z^{(i)T} z^{(j)}$$

subject to

$$\sum_{i=1}^{p} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \quad (i = 1, 2, \ldots, p). \tag{16.34}$$

Here $z^{(i)T} z^{(j)}$ is the inner product $< \phi(x^{(i)}), \phi(x^{(j)}) >$. This inner product is a scalar and is termed as a kernel function $K(x^{(i)}, x^{(j)})$ that maps the pair $(x^{(i)}, x^{(j)})$ to a real number. Figure 16.7 depicts the feature space and corresponding linear classifier.

Since the transformed patterns appear only as an inner product in the objective function, by employing the kernel function, we can solve (16.34) without explicitly computing $z^{(i)}$, i.e. without the explicit knowledge of the mapping ϕ. This novelty of the kernel approach has been one of the main reasons for the huge popularity and success of SVMs.

Definition 16.6.1. (Kernel Function). *A function* $K : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ *given by* $K(x^{(i)}, x^{(j)}) =< \phi(x^{(i)}), \phi(x^{(j)}) >$ *is called a Kernel function.*

In view of Definition 16.6.1, we can substitute $K(x^{(i)}, x^{(j)})$ for $z^{(i)T} z^{(j)}$ to get

$$\text{Max} \quad \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j K(x^{(i)}, x^{(j)})$$

subject to

$$\sum_{i=1}^{p} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad (i = 1, 2, \ldots, p). \tag{16.35}$$

Let $\overline{\alpha} = (\overline{\alpha_1}, \ldots, \overline{\alpha_p})$ be optimal to (16.35) and $S = \{i : \overline{\alpha_i} > 0\}$. Then $\overline{w} = \sum_{i \in S} \overline{\alpha_i} y_i z^{(i)}$. Now for a new pattern $x \in \mathbf{R}^n$, we find $f(x) = (\phi(x))^T \overline{w} - \overline{b}$ where $\overline{b} = -y_j + \phi(x^{(j)}))^T \overline{w}$ for some j such that $0 < \overline{\alpha}_j < C$. Therefore, substituting $\overline{w} = \sum_{i \in S} \overline{\alpha}_i y_i \phi(x^{(i)})$ and using the definition of the kernel K, we have

$$f(x) = \sum_{i \in S} \overline{\alpha}_i y_i K(x^{(i)}, x) - b,$$

where $S = \{i : \overline{\alpha_i} > 0\}$ and b is given by

$$b = -y_j + \sum_{i \in S} \overline{\alpha}_i \, y_i K(x^{(i)}, x^{(j)}),$$

for some j such that $0 < \overline{\alpha_j} < C$. In practise b may be chosen as the average of all such j.

Using $f(x)$, we assign label $+1$ to x if $f(x) > 0$ and -1 otherwise. It is important to note here that $f(x)$ does not require the function ϕ explicitly.

For the above methodology to work, we need to write $K(x, y) =< \phi(x), \phi(y) >$ for some $\phi : \mathbf{R}^n \to \mathcal{H}$, where $\mathcal{H}$ is a finite or infinite dimensional vector space equipped with the inner product (mostly we take $\mathcal{H} = \mathbf{R}^m (m > p)$). But is it always possible for any $K : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$? The following theorem is important in this context.

Theorem 16.6.1 (Mercer's Theorem). *Given a function $K : \mathbf{R}^n \times \mathbf{R}^n \longrightarrow \mathbf{R}$ there exists a mapping $\phi : \mathbf{R} \to \mathcal{H}$ and an expansion $K(x, y) = \phi(x)^T \phi(y)$ if and only if for every real valued function $g : \mathbf{R}^n \to \mathbf{R}$ with $\int (g(x))^2 dx < \infty$ we have*

$$\int K(x, y) \, g(x) \, g(y) \, dx \, dy \ \ge \ 0.$$

A function K satisfying the above condition is called a *Mercer kernel*. Therefore, in order to apply the kernel approach for SVMs, all we need to do is to choose a Mercer kernel. Some popular kernels are

(1) $K_q(x^{(1)}, x^{(2)}) = [1 + (x^{(1)})^T x^{(2)}]^q = [1 + q \, (x^{(1)})^T x^{(2)} + \ldots + ((x^{(1)})^T x^{(2)})]^q$. The kernel K_q as defined above, is called the *polynomial kernel of degree q*.

(2) $K_G(x^{(1)}, x^{(2)}) = \exp\left(\frac{-(x^{(1)} - x^{(2)})^T(x^{(1)} - x^{(2)})}{2\sigma^2}\right)$. The kernel K_G as defined above, is called the *Gaussian Kernel.*

Obviously the linear SVM studied in the previous sections corresponds to the *linear kernel* $K(x^{(1)}, x^{(2)}) = (x^{(1)})^T x^{(2)}$.

We now make another interesting observation. Given a kernel function K, neither the choice of $\mathcal{H}$ nor the choice of ϕ is unique. For example, consider $K(x, y) = (x^T y)^2$. Let $n = 2$. Then we can choose $\phi : \mathbf{R}^2 \to \mathbf{R}^3$ given by $\phi(a, b) = [a^2 , \sqrt{2}ab , b^2]$ for which $K(x, y) =< \phi(x), \phi(y) >$, $x, y \in \mathbf{R}^2$. Another choice of ϕ and $\mathcal{H}$ is $\phi_1 : \mathbf{R}^2 \to \mathbf{R}^4$ given by $\phi_1(a, b) = [a^2 , ab , ba , b^2]$. Here again $K_1(x, y) =< \phi_1(x), \phi_1(y) >$.

Example 16.6.1 *Let* $A = \{(-1, 0) , (1, 0)\}$ *and* $B = \{(0, 0)\}$ *be two sets in* $\mathbf{R}^2$ *such that class labels for* A *is* $+1$ *and for* B *it is* -1. *Show that* A *and* B *are not linearly separable. Use a suitable mapping* ϕ *to determine the classifier.*

Solution. The convex hulls of A and B are not disjoint and hence the problem is not linearly separable. Let us now take a mapping $\phi : \mathbf{R}^2 \to \mathbf{R}^3$ given by $\phi(a, b) = (a^2 , \sqrt{2}a , 1)$. Then, $\phi(-1, 0) = (1 , \sqrt{2} , 1)$, $\phi(0, 0) = (0 , 0 , 1)$, and $\phi(1, 0) = (1 , \sqrt{2} , 1)$ and these transformed patterns are linearly separable in $\mathbf{R}^3$. Therefore we can formulate the problem in $\mathbf{R}^3$ to get the hard margin classifier $\phi_3(a, b) = 4\phi_1(a, b)$. Therefore $\phi_3(a, b) = 4\phi_1(a, b)$ where $(a, b) \in \mathbf{R}^2$ and $\phi(a, b) = (\phi_1(a, b), \phi_2(a, b), \phi_3(a, b))$ means $1 = 4a^2$ which translates to the nonlinear decision function $4x^2 = 1$, i.e. the pair of straight lines $x = \pm\frac{1}{2}$ in the input space. We can also visualize this solution geometrically as given in Fig. 16.8.

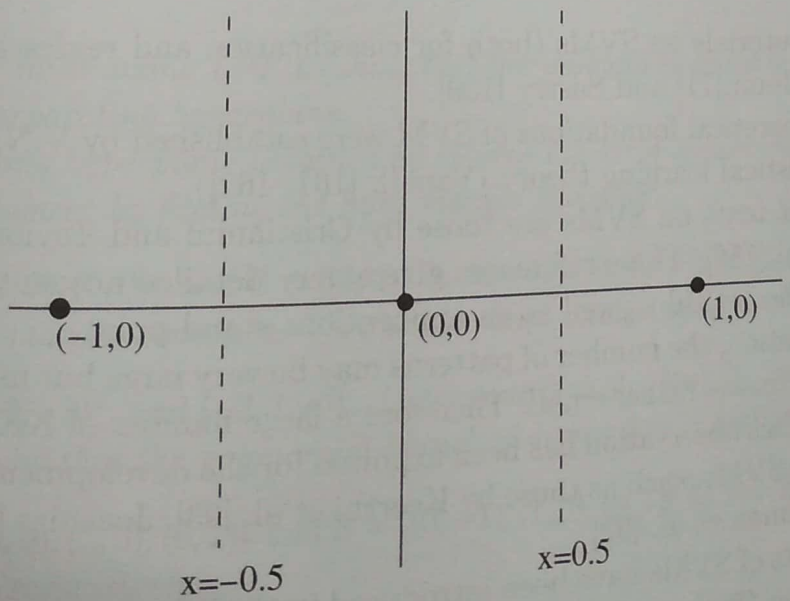


Fig. 16.8.

Remark 16.6.1 *The optimization problem (16.35) has the formulation of a soft margin classifier even though it is expected that in the feature space the problem will be linearly separable. This is because even after using the kernel K we can not be certain that strict separability has been fully achieved. Possibly the degree of separability is greater in the feature space than in the input space. In general, if the problem is linearly separable, or approximately so, it is better to employ a soft margin classifier to avoid overfitting.*

16.7 Summary and Additional Notes

- The contents of this chapter constitutes a very brief introduction to SVMs for binary data classification problems.
- Sections 16.2 to 16.5 are devoted to linear SVMs while nonlinear SVMs are discussed in section 16.6. Various issues related with Kernel SVMs are discussed and importance of Mercer theorem is emphasized.
- Two basic theoretical tools of mathematical programming, namely KKT conditions and duality, play a very major role in SVMs and this has been demonstrated time and again.
- Small numerical examples in $\mathbf{R}^2$ are discussed to clarify important concepts about SVMs.
- Though our discussion has been restricted to binary data classification only, SVMs have been applied to the multi class scenario as well.
- Support vector regression (SVR) is a SVM based approach to the problem of regression or function approximation. However, this topic has not been included in this chapter.
- Some useful tutorials on SVMs (both for classification and regression) are those by Burges [29], Gunn [71] and Sastry [139].
- The major theoretical foundations of SVM were established by V.N. Vapnik, and are based on statistical learning theory (Vapnik [161, 162]).
- Some excellent texts on SVMs are those by Cristianini and Taylor [42] and Taylor and Crisrianini [156]. These references give a very detailed account of kernel SVMs, from both a theoretical as well as an applications stand point.
- In a given situation, the number of patterns may be very large but usually the number of support vectors is rather small. Therefore a large number of KKT multipliers are usually zero. This observation has been expoited for the development of some efficient algorithms for SVMs, such as those by Keerthi et al. [93], Joachim [86], Mangasarian [111], and Kecman et al. [92].
- Several variants of SVMs have been introduced in the literature e.g. *proximal support vector machines* (Fung and Mangasarian [60]), *least squares support vector machines* (Suykens and Vandewalle [151]), *generalized eigen value proximal support vector machines* (Magasarian and Wild [112]), *knowledge based support vector machines* (Fung

et al. [62], Khemchandani et al. [95]) and *twin support vector machines* (Jayadeva et al.[83]). Some of these developments have been discussed in Lin and Wang [104] and Khemchandani [94].

- Some structured convex programming problems, e.g. *semidefinite programming* (SDP) and *second order cone programming* (SOCP) have been used in the SVM literature for the problem of optimal kernel selection. Some of the relevant references are Lanckriet et al. [100], Fung et al. [61] and Khemchandani [96].
- Some standard software packages are LIBSVM [41], SVMLIGHT[87].
- Benchmark data sets to demonstrate the efficiency of any new algorithmic development on SVM, are available at www.support-vector-machines.org and http://www.ics.uci.edu/ mlearn/MLRepository.html.

16.8 Exercises

16.1 *Let* $A = \{(0,0),(2,0),(0,2)\}$ *and* $B = \{(0,-1),(1,0),(0,1)\}$ *be two sets in* $\mathbf{R}^2$.

1. *Sketch the convex hulls of the sets A and B and hence conclude that these are not linearly separable.*
2. *Write the error minimizing LPP and solve the same by the simplex method. Identify the separating hyperplane and the associated misclassification error. Can you guess the answer geometrically?*
3. *Explicitly write the primal and dual QPPs to determine the soft margin classifier.*

16.2 *Let* $A = \{(-1,0),(0,0)\}$ *and* $B = \{(1,0)\}$ *be datasets with class labels +1 and -1, respectively.*

1. *Use the error minimizing LPP to show that the problem is linearly separable. Hence determine a separating hyperplane.*
2. *Is the separating hyperplane obtained by solving the error minimizing LPP optimal? Verify your answer by finding the hard margin classifier.*

16.3 *Let* $A = \{(2,0),(0,0)\}$ *and* $B = \{(4,4)\}$ *be two sets in* $\mathbf{R}^2$. *It is claimed that the line* $y = 3$ *is the hard margin classifier. Verify this claim analytically.*

16.4 *Let* $x = (0,2,-3)^T$ *and* $(-2,1,4)^T$. *Determine* $\|(x)_+\|_1$, $\|(y)_+\|_1$ *and* $\|(x+y)_+\|_1$ *and* $\|(x)_+ + (y)_+\|_1$. *Show that the polynomial kernel of degree 2 is a Mercer kernel.*

16.5 *Let* $A = \{(0,0),(2,0),(0,2)\}$ *and* $B = \{(0,-1),(1,0),(0,1)\}$ *be two datasets with class labels +1 and -1, respectively.*

1. *Use a polynomial kernel of degree 2 to find a soft margin classifier.*
2. *Write the optimization problem (both the primal and dual forms) if a Gaussian kernel with* $\sigma = 1$ *is employed for finding the soft margin classifier.*

17 Mathematical Programming Applications in Financial Mathematics

17.1 Introduction

Modern finance is a flavor of recent times. With exceptional contributions coming in from economists Kiyoshi Itō in 40's and 50's, Harry M. Markowitz in 50's, Fischer Black, Myron Scholes and Robert C. Merton in late 60's and early 70's, and many more, the subject of financial mathematics has witnessed an explosive growth. Financial mathematics focusses on applying mathematical or numerical techniques on the problems arising in financial economics. The necessity to understand the theoretical and computational aspects of the subject for the success of any organization makes it more appealing. Now a days the subject commands high level of attention among the researchers and students across the globe.

The aim of the chapter is to take a closer look at a particular problem of finance called *portfolio selection* from optimization viewpoint. The theory of optimal selection of portfolio was developed by H. M. Markowitz, a US economist, in 50's. He shared the Nobel prize in Economics with M. M. Miller and W. F. Sharpe in 1990 for his pioneer work in portfolio theory. Here, we present a brief description of the models and relate them to quadratic programming problems (QPP) through mean-variance analysis.

17.2 Technical Terminologies

Let us first get familiar with some terminologies often used in the chapter.

Definition 17.2.1 (Asset). *An asset is a valuable economic entity from which the future economic benefits are expected to flow to the owner of the asset. It is controlled by the owner who legally acquired it as a result of past transactions or other events.*

Assets can be classified in many categories. Here we list only few of them. Physical or tangible assets are fixed assets like real estate, machinery, furniture etc. These assets are

also known as capital assets in accounting. Intangible assets are defined as those non-monetary assets that cannot be physically measured like patents, goodwill, competitive knowledge etc. Liquid or financial assets include cash, bonds, shares, mutual funds, currency etc., gold and other precious metals are assets that are both tangible and liquid.

Throughout the chapter, an asset always means the financial asset only.

Definition 17.2.2 (Return). *The return on an asset is an indicator of gain/loss in the investment of an asset in the financial market. It is determined by the following formula*

$$\text{return} = \frac{\text{amount received} - \text{amount invested}}{\text{amount invested}}.$$

Suppose that the current price of an asset is $A(0)$ and after $\mathbf{T}$ time period the asset is sold off at an amount equals $A(\mathbf{T})$. Then the return on an asset for $\mathbf{T}$ time period is given by

$$r = \frac{A(\mathbf{T}) - A(0)}{A(0)}.$$

The positive value of return on an asset signifies gain while the negative return signifies loss, zero return means neither gain nor loss from the investment.

Remark 17.2.1 *It is important to mention here that definition 17.2.2 is given in percentage and hence return on an asset is actually to be understood as a rate of return on an asset. However, we shall continue to call it as return on an asset for consistency with the financial market glossary.*

Definition 17.2.3 (Risk). *The risk is often defined as the degree of uncertainty of return on an asset. It signifies the possibility of loss in the investment.*

The risk can either be zero, implying that the asset is risk-free, or positive, implying the asset is risky. If the asset is risk-free then the future value of the asset is known with certainty otherwise the future value of the risky asset is uncertain. The financial asset can thus be classified as risk-free asset like bond or fixed deposit and risky asset like share or mutual fund.

There are two kinds of risks associated with a risky asset viz., systematic risk and unsystematic risk.

(i) The systematic risk is common to all business. It is concerned with the institutional structure of the banking system, money and capital markets, credit and fiscal policies and economic policies which govern the market economy.

(ii) The unsystematic risk, also called the diversifiable risk or residual risk, is unique to a company such as, workers strike, the outcome of unfavorable litigation, sudden discovery of deficiencies in a product of a company, a natural catastrophe etc. Risks of

this nature can be eliminated through diversification by investing the original amount of money in more than one asset.

Definition 17.2.4 (Short Selling). *It refers to a situation where an investor actually does not own an asset but he establishes a market position by selling an asset in anticipation that the price of that asset will fall. We say that the investor has taken a short position. Mathematically, this situation can be explained by taking the number of assets owned by the investor as negative.*

(i) A short position in risk-free asset simply means borrowing cash from the market at some interest rate or borrowing rate, repaying that loan along with the interest at later stage and close the short position.

(ii) In case of risky asset, a short position is realized by short selling. Here, an investor borrows an asset, sells it at some price, say S_0. The short position is closed when the investor buys back the asset at, say S_1, and returns it to the owner. The difference $S_0 - S_1$ is the gain/loss of the investor.

Remark 17.2.2 *Short selling is generally considered to be very risky. If the price S_1 of an asset in the short position of an investor shoots up than the investor stands to loose enormously. Many perceive short selling to be a cause of market crashes and hence short selling is prohibited in certain markets but of course it is not completely forbidden. However, the very notion that, a short seller can cause a permanent fall in the share prices, itself is debateable for any security which is short sold is to be bought back and hence there is no permanent supply of the shares in the market by the short sellers. Many economists strongly believes that banning short selling does more harm than good to the market as in that situation the market prices are controlled by the alleged manipulators and irrational investors. Till some time ago, in India, short-selling was only available to retail investors (an individual who purchases small amounts of securities for himself/herself, also called small investor) and not allowed to the institutional investors (entity with large amounts to invest, such as investment companies, mutual funds, brokerages, insurance companies, pension funds, investment banks and endowment funds). But very recently guidelines have been issued by securities and exchange board of India (SEBI) that enable institutional investors to sell stocks without owning them under certain rules.*

Remark 17.2.3 *If the number of assets of a particular kind owned by the investor is positive then the investor is said to have taken a long position.*

In order to build a mathematical model of the real world financial scenario we make certain assumptions.

1. The prices of all assets at any time are strictly positive.
2. The return r on an asset is a random variable taking finitely many values.

3. An investor can own a fraction of an asset. This assumption is known as *divisibility*.
4. An asset can be bought or sold on demand in any quantity at the market price. This assumption is known as *liquidity*.
5. There are no commissions/transaction costs.

We are now in a position to move towards our main aim, i.e. *portfolio optimization*. We first define a portfolio and then consider *two-asset* and *multi-asset* portfolio theories in the subsequent sections.

Definition 17.2.5 (Portfolio). *A portfolio is a collection of two or more assets, say, $a_1, \ldots, a_n$, represented by an ordered n-tuple $\Theta = (x_1, \ldots, x_n)$, where $x_i \in \mathbf{R}$ is the number of units of the asset a_i $(i = 1, \ldots, n)$ owned by the investor.*

We consider only a single period model, that is, in between the initial time taken as $t = 0$ and the final transaction time taken as $t = \mathbf{T}$, no transaction ever takes place.

Let $V_i(0)$ and $V_i(\mathbf{T})$ be the values of the i-th asset at $t = 0$ and $t = \mathbf{T}$, respectively. Let $V(0)$ and $V(\mathbf{T})$ denote the values of the portfolio $\Theta = (x_1, \ldots, x_n)$ at $t = 0$ and $t = \mathbf{T}$, respectively. Then, we have

$$V(0) = \sum_{i=1}^{n} x_i V_i(0),$$

$$V(\mathbf{T}) = \sum_{i=1}^{n} x_i V_i(\mathbf{T}).$$

Definition 17.2.6 (Asset Weights). *The weight w_i of the asset a_i is the percentage of the value of the asset in the portfolio at $t = 0$, i.e.*

$$w_i = \frac{x_i V_i(0)}{\sum_{i=1}^{n} x_i V_i(0)} \qquad (i = 1, \ldots, n).$$

It can be observed that $w_1 + \cdots + w_n = 1$.

Remark 17.2.4 *In a portfolio, if $w_i < 0$, for some i, it indicates that the investor has taken a short position on the i-th asset a_i.*

Let r_i be the return (definition 17.2.2) on the i-th asset. Then

$$r_i = \frac{V_i(\mathbf{T}) - V_i(0)}{V_i(0)} \qquad (i = 1, \ldots, n).$$

Suppose the portfolio comprises of two assets a_1 and a_2 with initial prices $V_1(0)$ =Rs 25 and $V_2(0)$ =Rs 50. We purchase 20(= x_1) units of a_1 and 15(= x_2) units of a_2. Then the

initial worth of portfolio is $V(0) = x_1V_1(0) + x_2V_2(0) =$ Rs 1250. The proportion of allocation of Rs 1250 between two assets is $w_1 = \frac{(25)(20)}{1250} = 40\%$ and $w_2 = \frac{(50)(15)}{1250} = 60\%$. After one year, suppose the assets return values are $V_1(1) =$Rs 30 and $V_2(1) =$Rs 45. Then the portfolio worth would be $V(1) = (20)(30) + (15)(45) =$Rs 1275. The return on a_1 is $r_1 = \frac{30-25}{25} = 20\%$, and $r_2 = \frac{45-50}{50} = -25\%$.

Due to our assumption, the return value of any asset is a random variable and thus we can talk about the expected value μ_i of the return r_i, i.e.

$$\mu_i = E(r_i) \qquad (i = 1, \ldots, n).$$

The risk associated with the asset is given by the variance of the return, i.e.

$$\sigma_i^2 = var(r_i) \qquad (i = 1, \ldots, n).$$

Example 17.2.1 *Suppose there are two investment opportunities O_1 and O_2 giving the following returns in two different market scenarios*

scenario	*probability of scenario*	*return* O_1	*return* O_2
ω_1	0.25	8%	12%
ω_2	0.75	11%	9%

Which of the two is a more risky opportunity?

Solution Here

$$\mu_1 = E(O_1) = (0.25)(0.08) + (0.75)(0.11) = 0.1025,$$
$$\mu_2 = E(O_2) = (0.25)(0.12) + (0.75)(0.09) = 0.0975.$$

$$\sigma_1^2 = var(O_1) = (0.25)(0.08 - 0.1025)^2 + (0.75)(0.11 - 0.1025)^2 = 0.00016875,$$
$$\sigma_2^2 = var(O_2) = (0.25)(0.12 - 0.0975)^2 + (0.75)(0.09 - 0.0975)^2 = 0.00016875.$$

Thus both opportunities are equally risky although the expected return on the first opportunity (10.25%) is higher than that on the second opportunity (9.75%).

17.3 Two Asset Portfolio Optimization

Consider a portfolio with two assets, say, a_1, a_2 with weights w_1, w_2, returns r_1, r_2 and standard deviations σ_1, σ_2, respectively. Then the portfolio expected return μ and portfolio variance σ are respectively given by

$$\mu = E(w_1r_1 + w_2r_2) = w_1\mu_1 + w_2\mu_2, \tag{17.1}$$
$$\sigma^2 = var(w_1r_1 + w_2r_2) = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2\rho w_1w_2\sigma_1\sigma_2. \tag{17.2}$$

Here ρ is the coefficient of correlation between r_1 and r_2, and the value of ρ lies in $[-1, 1]$. We pause here to analyze the effect of ρ on the risk involved in a portfolio.

What we shall be observing is that the value of ρ provides a measure of the extent of diversification of portfolio possible to reduce risk. The more negative the value of ρ, the greater are the benefits of the portfolio diversification.

As w_1 and w_2 are weights representing the proportions of total investment in two assets a_1 and a_2, respectively, we have $w_1 + w_2 = 1$. Moreover, in case of short selling, the weights can be negative. Subsequently, we write $w_1 = 1 - s$, and so, $w_2 = s$, $s \in \mathbf{R}$. Now, it follows from relations (17.1) and (17.2) that

$$\mu = (1 - s)\mu_1 + s\mu_2, \tag{17.3}$$

$$\sigma^2 = (1 - s)^2\sigma_1^2 + s^2\sigma_2^2 + 2\rho(1 - s)s\sigma_1\sigma_2. \tag{17.4}$$

Relation (17.4) can be simplified as

$$\sigma^2 = (\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2)s^2 - 2\sigma_1(\sigma_1 - \rho\sigma_2)s + \sigma_1^2. \tag{17.5}$$

Without loss of generality we assume that $0 < \sigma_1 \leq \sigma_2$. We discuss the following two independent cases.

$$(i) \quad \rho = \pm 1, \qquad\qquad (ii) \quad -1 < \rho < 1.$$

Case (i). $\rho = \pm 1$. From relations (17.3) and (17.5), the portfolio expected return and variance are respectively given by

$$\mu = (1 - s)\mu_1 + s\mu_2,$$
$$\sigma = |(1 - s)\sigma_1 \pm s\sigma_2|.$$

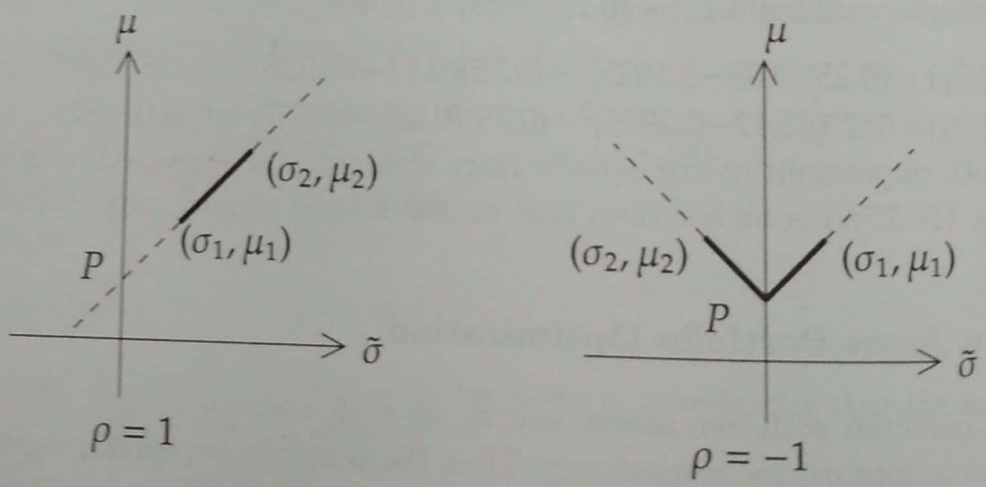


Fig. 17.1.

For $s \in [0,1]$ both weights are non-negative and thus the portfolio has no short positions. If $s > 1$ then $w_1 < 0$, it means that asset a_1 is held short, while if $s < 0$ then $w_2 < 0$ and it indicates that asset a_2 is held short. It may be noted that an investor can not take short position on both the assets simultaneously.

For $\mu = (1-s)\mu_1 + s\mu_2$, $\tilde{\sigma} = (1-s)\sigma_1 + s\sigma_2$, we plot the $(\tilde{\sigma}, \mu)$-graph, illustrated in the first graph of Fig 17.1. The second graph in Fig 17.1 corresponds to the case when $\mu = (1-s)\mu_1 + s\mu_2$, $\tilde{\sigma} = (1-s)\sigma_1 - s\sigma_2$, The bold parts in the graphs correspond to $s \in [0,1]$.

Subsequently, we plot the standard deviation-mean diagram of the portfolio, i.e. the (σ, μ)-graph, for $\rho = \pm 1$. Observe that $\sigma = |\tilde{\sigma}|$. The two graphs are depicted in Fig 17.2.

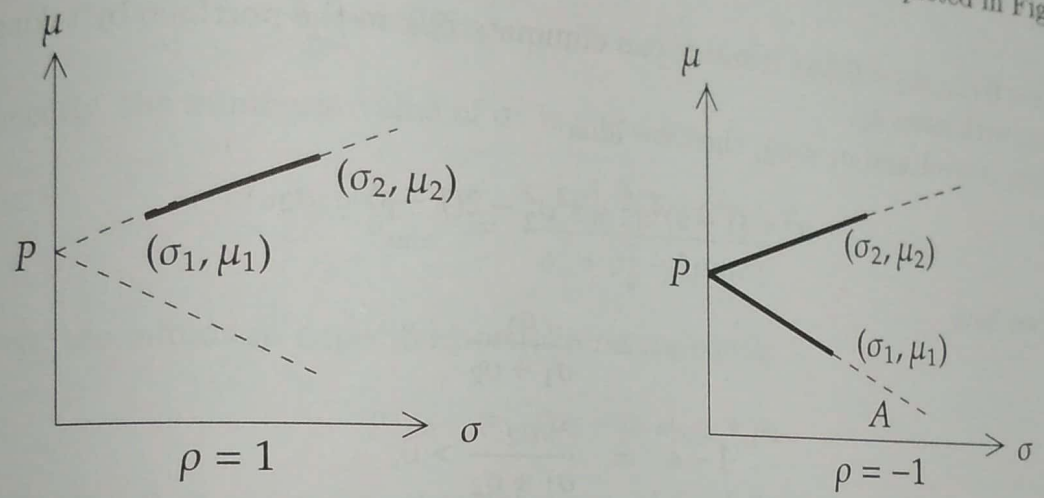


Fig. 17.2.

From Fig 17.2 we have the following observations to share.

Remark 17.3.1 *(i) When the returns of the two assets are perfectly positively correlated, i.e. $\rho = 1$, the higher expected return of the portfolio comes along with the higher risk. Furthermore, the risk of the portfolio can be completely eliminated by taking a short position on asset a_2 (point P in the first figure of Fig 17.2).*
(ii) When the returns of the two assets are perfectly negatively correlated, i.e. $\rho = -1$, the expected portfolio return increases with the gradual decrease in the overall risk of the portfolio (point A to point P in the second figure of Fig 17.2) till the risk is completely eliminated (point P). After that the higher return comes with higher risk as weight of riskier asset increases.

Below we provide mathematical justification of the above observations.
When $\rho = 1$ and $\sigma_1 = \sigma_2$, then $\sigma_{\min} = \sigma_1$.
When $\rho = 1$ and $\sigma_1 < \sigma_2$, then we have

$$\sigma^2 = (1-s)^2\sigma_1^2 + s^2\sigma_2^2 + 2(1-s)s\sigma_1\sigma_2. \tag{17.6}$$

Our aim is to minimize σ, or equivalently σ^2. Now

$$\frac{d\sigma^2}{ds} = s(\sigma_1 - \sigma_2)^2 - \sigma_1(\sigma_1 - \sigma_2),$$

$$\frac{d^2\sigma^2}{ds^2} = (\sigma_1 - \sigma_2)^2 > 0.$$

Thus, to minimize the risk σ, we must choose the weight s such that $\frac{d\sigma^2}{ds} = 0$. Thereby (17.6) yield

$$s = \frac{\sigma_1}{\sigma_1 - \sigma_2} < 0,$$

$$\mu_{\min} = \frac{\sigma_1\mu_2 - \sigma_2\mu_1}{\sigma_1 - \sigma_2}.$$

Since $s < 0$ (i.e. $w_2 < 0$), an investor can eliminate risk in the portfolio by taking a short position with asset a_2.

When $\rho = -1$ and $\sigma_1 < \sigma_2$, then we have

$$\sigma^2 = (1-s)^2\sigma_1^2 + s^2\sigma_2^2 - 2(1-s)s\sigma_1\sigma_2.$$

Thus, we have

$$s = \frac{\sigma_1}{\sigma_1 + \sigma_2} > 0,$$

$$1 - s = \frac{\sigma_2}{\sigma_1 + \sigma_2} > 0,$$

$$\mu_{\min} = \frac{\sigma_1\mu_2 + \sigma_2\mu_1}{\sigma_1 + \sigma_2}.$$

Since $s > 0$ and $1 - s > 0$ hence the investor can eliminate the risk in the portfolio without restoring to short selling.

Case (ii). We now consider the second case when $-1 < \rho < 1$.

Recall relations (17.3) and (17.5), we have

$$\mu = (1-s)\mu_1 + s\mu_2,$$
$$\sigma^2 = (\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2)s^2 - 2\sigma_1(\sigma_1 - \rho\sigma_2)s + \sigma_1^2. \tag{17.7}$$

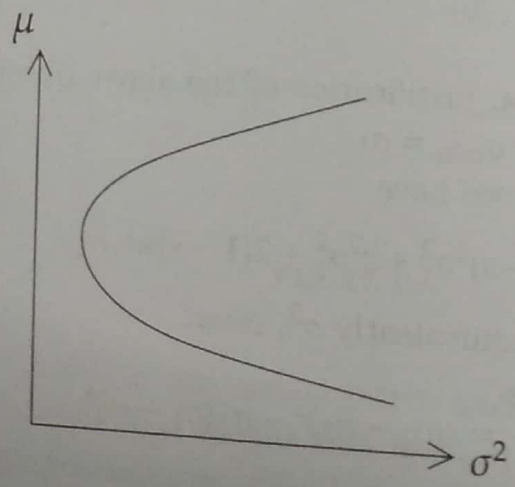


Fig. 17.3.

(17.7) is a quadratic equation in s representing a parabola depicted in Fig 17.3. We wish to minimize σ^2. Now,

$$\frac{d\sigma^2}{ds} = 0 \Rightarrow s = \frac{\sigma_1^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}.$$

And

$$\frac{d^2\sigma^2}{ds^2} = 2((\sigma_1 - \rho\sigma_2)^2 + \sigma_2^2(1-\rho^2)) > 0.$$

Consequently, the minimum value of σ^2 is given by

$$\sigma^2_{\min} = \frac{\sigma_1^2\sigma_2^2(1-\rho^2)}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2},$$

Moreover, the minimum expected portfolio return equals

$$\mu_{\min} = (\mu_2 - \mu_1)s_{\min} + \mu_1.$$

Remark 17.3.2 *It is important to take note of the following points.*

(i) The condition $-1 \leq \rho \leq \frac{\sigma_1}{\sigma_2}$ is equivalent to $0 < s_{\min} < 1$. Thus the minimum risk can be achieved without short selling. Also,

$$\sigma^2_{\min} = 0 \Leftrightarrow \rho = -1.$$

(ii) $\rho = \frac{\sigma_1}{\sigma_2} \Leftrightarrow s_{\min} = 0 \Leftrightarrow \sigma^2_{\min} = \sigma_1^2$.

(iii) The condition $\frac{\sigma_1}{\sigma_2} < \rho \leq 1$ is equivalent to $s_{\min} < 0$. In this case the investor has taken a short position on asset a_2 in order to minimize the portfolio risk. Further,

$$\sigma^2_{\min} = 0 \Leftrightarrow \rho = 1.$$

The entire theory of this section is summarized in Fig 17.4.

The risk-return relation of two assets for various values of ρ provides us with a triangle ΔAPB. The points A and B signify undiversified portfolios. Since $-1 \leq \rho \leq 1$, ΔAPB specifies the limit of diversification. The risk-return relation for all values of ρ except ± 1 lie within this triangle.

Let us verify the two-assets portfolio theory by considering the following example and plotting the corresponding (σ, μ)-graph.

Let A and B be two assets with expected returns 12% and 16% and standard deviation 16% and 20%, respectively. Let x_A and x_B denote the number of units of asset A and asset B, respectively, in a portfolio.

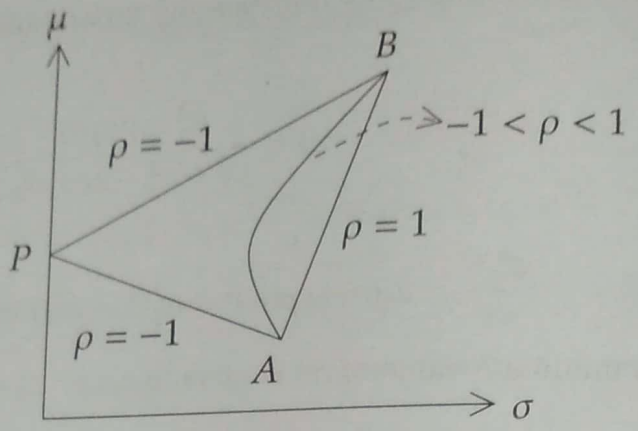


Fig. 17.4.

x_A	x_B	μ	σ				
			$\rho = 1$	$\rho = 0.5$	$\rho = 0$	$\rho = -0.5$	$\rho = -1$
100	0	12.00	16.00	16.00	16.00	16.00	16.00
90	10	12.40	16.40	16.50	14.54	13.51	12.40
80	20	12.80	16.80	15.20	13.41	11.34	8.80
70	30	13.20	17.20	15.12	12.71	9.71	5.20
60	40	13.60	17.60	15.26	12.50	8.91	1.60
50	50	14.00	18.00	15.62	12.81	9.17	2.00
40	60	14.40	18.40	16.18	13.60	10.40	5.60
30	70	14.80	18.80	16.92	14.80	12.32	9.20
20	80	15.20	19.20	17.82	16.32	14.66	12.80
10	90	15.60	19.60	18.85	18.07	17.26	16.40
0	100	16.00	20.00	20.00	20.00	20.00	20.00

Remark 17.3.3 *From the above table we can easily observe the following.*

(i) The two assets can be combined in such a way that the portfolio risk is less than the individual risks. For instance, when the assets are taken in the ratio 80 : 20 *and* $\rho = 0.5$ *then the risk in the portfolio is* 15.20% *whereas if the entire investment is put in asset A only or in asset B only then the risks involved are* 16% *and* 20%*, respectively. While if* $\rho = -0.5$ *then* 60 : 40 *ratio can bring down the risk to* 8.91%*. It signifies that in many circumstances diversification of the portfolio is advisable for reducing the risk. The underline principle is thus that 'do not put all the eggs in one basket'.*

(ii) For a given weight combination the risk reduces as ρ *moves from 1 to -1, i.e. the more uncorrelated the assets are the better is the risk-return relation.*

(iii) When $\rho < 1$ *then certain combinations of the assets are better than the others. For*

example, for $\rho = 0$ the 60:40 combination with $\mu = 13.60\%$ and $\sigma = 12.50\%$ is better than the 70:80 combination with $\mu = 13.20\%$ and $\sigma = 12.71\%$. For $\rho = 0.5$, 70 : 30 combination yields return 13.20% and risk 15.12% which is better than the 80 : 20 ratio that gives lower return 12.80% with higher risk 15.20%.

From the above discussion we can thus conclude that by choosing appropriate ratio of investment between two assets, the risk can be reduced considerably.

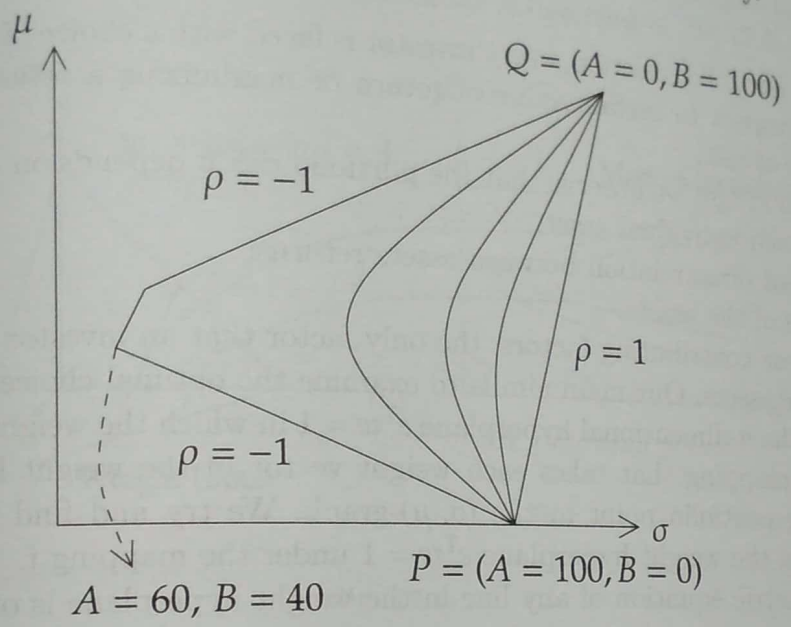


Fig. 17.5.

In Fig 17.5, as the weights of the assets A and B vary from 100 to 0 and 0 to 100, respectively, the curve between risk and return moves from extreme right (south-east point P) to extreme left and turns to move from extreme left to extreme right (north east point Q) corresponding to the various values of ρ between -1 and 1.

17.4 Multi Asset Portfolio Optimization

In this section we extend the two-asset portfolio theory to n - asset portfolio theory.

The weights of the various assets $a_1, \ldots, a_n$ in the portfolio are written in the vector form $w = col(w_1, \ldots, w_n)$. Let $e^T = (1, \ldots, 1) \in \mathbf{R}^n$. Then $w_1 + \cdots + w_n = 1$ can be expressed as $e^T w = 1$. Let $m^T = (\mu_1, \ldots, \mu_n)$ be the expected return vector of the portfolio, where, $\mu_i = E(r_i)$ $(i = 1, \ldots, n)$, and $C = [c_{ij}]$ denotes the $n \times n$ variance-covariance matrix with entries $c_{ij} = \text{cov}(r_i, r_j)$ $(i, j = 1, \ldots, n)$. Note that $c_{ii} = \sigma_i^2$ $(i = 1, \ldots, n)$. Obviously C is a symmetric matrix. Also, C is a positive definite matrix and thus it is invertible.

Now the expected return μ of the portfolio is given by

$$\mu = E\left(\sum_{i=1}^{n} w_i r_i\right) = \sum_{i=1}^{n} w_i \mu_i = m^T w,$$

and the risk σ^2 of the portfolio is

$$\sigma^2 = var\left(\sum_{i=1}^{n} w_i r_i\right) = \sum_{i,j=1}^{n} c_{ij} w_i w_j = w^T C w. \tag{17.8}$$

During a portfolio selection, every investor is faced with a choice of either minimizing a risk with respect to certain value of return or maximizing a return with respect to certain value of risk.

Now, from (17.8), we observe that the portfolio risk σ depends on three factors, viz.,

(i) risk of each individual asset;
(ii) coefficient of correlation between assets returns;
(iii) weights of the assets.

Out of these contributing factors, the only factor that an investor can control is the weights of the assets. Our main aim is to examine the optimal choice of these weights.

Consider the n-dimensional hyperplane $e^T w = 1$ in which the weight vector w resides. Let f be the mapping that takes each weight vector in the weight hyperplane to the corresponding portfolio point in the (σ, μ)-graph. We try and find the image of any straight line in the weight hyperplane $e^T w = 1$ under the mapping f.

The parametric equation of any line in the weight hyperplane is of the form

$$\begin{aligned} l(\xi) &= (s_1\xi + b_1, \ldots, s_n\xi + b_n) \\ &= \xi s + b, \qquad -\infty < \xi < \infty \end{aligned}$$

where $s = (s_1, \ldots, s_n)$ and $b = (b_1, \ldots, b_n)$. Let w be any point on this line. Then

$$\begin{aligned} \mu &= m^T w \\ &= m^T(\xi s + b) \\ &= \xi(m^T s) + (m^T b). \end{aligned}$$

Let $\alpha = (m^T s)^{-1}$, $\beta = -(m^T b)(m^T s)^{-1}$. Then, $\xi = \mu\alpha + \beta$. Moreover,

$$\begin{aligned} \sigma^2 &= w^T C w \\ &= (\xi s + b)^T C(\xi s + b) \\ &= (s^T C s)\xi^2 + (s^T C b + b^T C s)\xi + b^T C b \\ &\equiv \gamma\xi^2 + \delta\xi + \eta. \end{aligned}$$

Substituting the value of ξ we get

$$\sigma^2 = \gamma(\alpha\mu + \beta)^2 + \delta(\alpha\mu + \beta) + \eta. \tag{17.9}$$

As ξ varies from $-\infty$ to ∞, the ordered pair (σ^2, μ) traces out a parabola given by (17.9) which lies in (σ, μ)- graph with axis parallel to σ-axis and sides open on the right. We are actually interested in (σ, μ)-graph. Taking the square root of σ^2, the resulting curve is

$$\sigma = \sqrt{\gamma(\alpha\mu + \beta)^2 + \delta(\alpha\mu + \beta) + \eta}. \tag{17.10}$$

This curve is called a *Markowitz curve*. Thus, each line in the weight hyperplane is mapped onto a Markowitz curve. This phenomena is depicted in Fig 17.6.

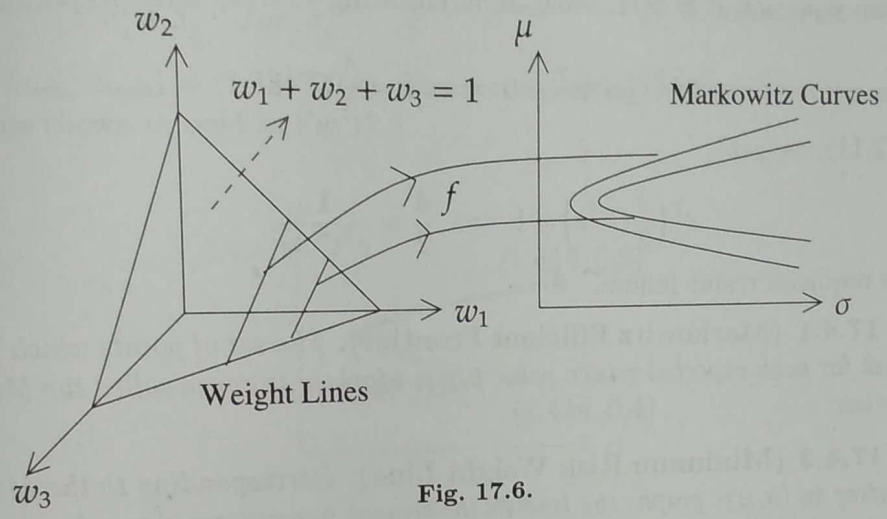


Fig. 17.6.

Remark 17.4.1 *Here it is important to note that the Markowitz curve (17.10) is not a parabola. In fact the main difference between the parabola (17.9) and the Markowitz curve (17.10) in (σ, μ)-graph is that a tangent can be drawn to the parabola (17.9) from any point on the μ-axis, whereas the Markowitz curve behaves almost as a straight line as $\mu \to \infty$, thereby, it is not possible to draw a tangent to the Markowitz curve as $\mu \to \infty$. This difference may not sound significant right now but it plays a vital role when the portfolio consists of one risk-free asset. We shall be addressing to this type of portfolio in the next section. For the current discussion we have assumed that all the assets in the portfolio are risky.*

As already observed that among all the factors affecting the risk σ^2 of the portfolio, the only factor that can be controlled by an investor is the weight vector w. In other words it is same as saying that an investor can decides upon appropriate weight vector to minimize the overall risk in the investment.

Theorem 17.4.1 *A portfolio with minimum risk has weights given by*

$$w = \frac{C^{-1}e}{e^T C^{-1} e}.$$

Proof. The problem is to

$$\begin{aligned} &\text{Min} && \sigma^2 = w^T C w \\ &\text{subject to} \\ & && e^T w = 1. \end{aligned} \tag{17.11}$$

Using the Lagrange multiplier $\lambda \in \mathbf{R}$, we minimize the Lagrangian

$$L(w, \lambda) = w^T C w + \lambda(1 - e^T w). \tag{17.12}$$

Note that λ is unrestricted in sign because the constraint in the risk minimization problem is an equation $e^T w = 1$. Now, differentiating (17.12) with respect to w, we obtain

$$2w^T C - \lambda e^T = 0 \implies w = \frac{\lambda}{2} C^{-1} e.$$

Using (17.11), we get

$$e^T \left(\frac{\lambda}{2} C^{-1} e\right) = 1 \implies \frac{\lambda}{2} = \frac{1}{e^T C^{-1} e}.$$

Thus the requisite result follows. □

Definition 17.4.1 (Markowitz Efficient Frontier). *The set of points which provides minimum risk for each expected return value μ is a Markowitz curve called the Markowitz efficient frontier.*

Definition 17.4.2 (Minimum Risk Weight Line). *Corresponding to the Markowitz efficient frontier in (σ, μ)- graph, the line in the weight hyperplane $e^T w = 1$ is called the minimum risk weight line.*

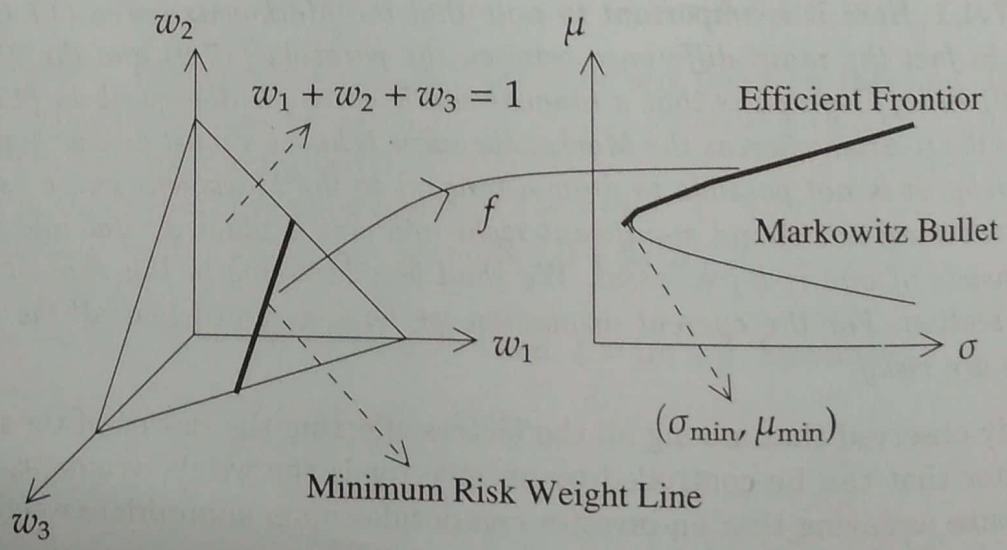


Fig. 17.7.

The Markowitz efficient frontier and the minimum risk weight line are depicted in Fig 17.7.

Example 17.4.1 *Suppose there are two assets a_1 and a_2 with $\mu_1 = 0.4$, $\mu_2 = 0.8$, $\sigma_1^2 = 2 = \sigma_2^2$, $\sigma_{12} = 1$. Obtain the minimum variance point and sketch the entire efficient frontier.*

Solution Here, we can note that

$$\rho = \frac{\sigma_{12}}{\sigma_1\sigma_2} = \frac{1}{2}; \qquad \sigma^2_{\min} = \frac{\sigma_1^2\sigma_2^2(1-\rho^2)}{\sigma_1^2+\sigma_2^2-2\rho\sigma_1\sigma_2} = \frac{3}{2};$$

$$s_{\min} = \frac{\sigma_1^2-\rho\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2-2\rho\sigma_1\sigma_2} = \frac{1}{2}; \qquad \mu_{\min} = (\mu_2-\mu_1)s_{\min}+\mu_1 = \frac{3}{5}.$$

Thus $(\sigma_{\min}, \mu_{min}) = (1.2247, 0.6)$. The corresponding Markowitz curve and efficient frontier are shown in bold in Fig 17.8.

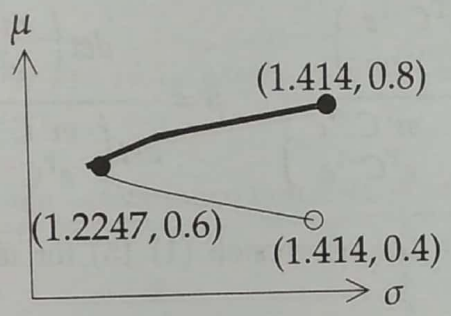


Fig. 17.8.

In many cases, it is more likely that an investor provides a fixed value of the expected return, say μ, that he desired to achieve. He has to decide the right investment strategy to obtain the return μ with the minimum risk. We look at this scenario in the result to follow.

Theorem 17.4.2 *For a given expected return μ, the portfolio with minimum risk has weights given by*

$$w = \frac{det\begin{pmatrix} \mu & m^TC^{-1}e \\ 1 & e^TC^{-1}e \end{pmatrix}C^{-1}m + det\begin{pmatrix} m^TC^{-1}m & \mu \\ e^TC^{-1}m & 1 \end{pmatrix}C^{-1}e}{det\begin{pmatrix} m^TC^{-1}m & m^TC^{-1}e \\ e^TC^{-1}m & e^TC^{-1}e \end{pmatrix}}. \tag{17.13}$$

Proof. We wish to solve the following quadratic programming problem

$$\text{Min} \quad \sigma^2 = w^TCw$$

subject to

$$\begin{aligned} m^Tw &= \mu \\ e^Tw &= 1. \end{aligned} \tag{17.14}$$

This is a convex quadratic programming problem with unrestricted variable vector w. Using the Lagrange multipliers $\alpha, \beta \in \mathbf{R}$, we minimize the Lagrangian

$$L(w, \alpha, \beta) = w^T C w + \alpha(\mu - m^T w) + \beta(1 - e^T w)$$

to obtain

$$2w^T C - \alpha m^T - \beta e^T = 0 \implies w = \frac{1}{2} C^{-1}(\alpha m + \beta e). \tag{17.15}$$

Substituting the value of w in (17.14), we get

$$\begin{aligned} (m^T C^{-1} m)\alpha + (m^T C^{-1} e)\beta &= 2\mu \\ (e^T C^{-1} m)\alpha + (e^T C^{-1} e)\beta &= 2. \end{aligned}$$

Solving the above two equations for α and β, we obtain

$$\alpha = \frac{det\begin{pmatrix} \mu & m^T C^{-1} e \\ 1 & e^T C^{-1} m \end{pmatrix}}{det\begin{pmatrix} m^T C^{-1} m & m^T C^{-1} e \\ e^T C^{-1} m & e^T C^{-1} e \end{pmatrix}}, \quad \beta = \frac{det\begin{pmatrix} m^T C^{-1} m & \mu \\ e^T C^{-1} m & 1 \end{pmatrix}}{det\begin{pmatrix} m^T C^{-1} m & m^T C^{-1} e \\ e^T C^{-1} m & e^T C^{-1} e \end{pmatrix}}$$

Substituting these values in the expression (17.15) for w , we get the required expression (17.13). □

Definition 17.4.3 (Markowitz Bullet). *Since the efficient frontier contains all the points of minimum risk for the preassigned value of return, therefore all the feasible points of problem (17.14) lie on or to the right of this frontier. Due to its shape, this region is called the Markowitz Bullet.*

It is worthwhile to pause here to make an important observation about the minimum-variance set for a fixed return. Recall from (17.14) and (17.15) that, for a given value of return μ, the points of minimum variance must satisfy the following system of $(n+2)$ linear equations in $(n+2)$ unknowns $w \in \mathbf{R}^n$, $\alpha \in \mathbf{R}$, $\beta \in \mathbf{R}$.

$$\begin{aligned} 2w^T C - \alpha m^T - \beta e^T &= 0 \\ m^T w &= \mu \\ e^T w &= 1. \end{aligned} \tag{17.16}$$

Suppose we solve the system (17.16) for two distinct values of expected return μ, say $\bar{\mu}^1$ and $\bar{\mu}^2$. Let the two solutions be $(w^1 = (w_1^1, \ldots, w_n^1)^T, \alpha^1, \beta^1)^T$ and $(w^2 = (w_1^2, \ldots, w_n^2)^T, \alpha^2, \beta^2)^T$, respectively. Then it is simple to verify that the combination portfolio, $\lambda(w^1, \alpha^1, \beta^1)^T + (1-\lambda)(w^2, \alpha^2, \beta^2)^T$, $\lambda \in \mathbf{R}$, is also a solution of the system (17.16) corresponding to the return $\lambda\bar{\mu}^1 + (1-\lambda)\bar{\mu}^2$. Therefore, in order to solve (17.16) for every value of μ, one is only required to solve it for two distinct values of μ and

then form the combination of the two solutions. Thus, the knowledge of two distinct portfolios yielding the minimum variances is sufficient to generate the entire minimum variance set. This result is significant from investor point of view. Also, it demonstrates a good application of KKT optimality conditions. The result is known as the *two fund theorem.*

Theorem 17.4.3 (Two Fund Theorem). *Two efficient portfolios can be established so that any other efficient portfolio can be duplicated, in terms of mean and variance, as a linear combination of these two assets. In other words, it says that, an investor seeking an efficient portfolio need to invest only in the combination of these two assets.*

The most convenient way to get two solutions of (17.16) is to assign two distinct values to α and β, and then work out the solutions. The most convenient choices are $\alpha = 1$, $\beta = 0$ and $\alpha = 0$, $\beta = 1$. The above discussion is illustrated through the following example.

Example 17.4.2 *Consider three risky assets with the covariance matrix and expected returns as follows.*

variance - covariance matrix(C)			return(M)
2	1	0	0.4
1	2	1	0.8
0	1	2	0.8

Find two portfolios yielding the minimum variance. Also, determine the expected returns from these two portfolios. Using the two fund theorem, construct the portfolio giving the return of 33.4% *with minimum risk.*

Solution Taking $\alpha = 0$, $\beta = 1$ in (17.16), we need to solve: $\sum_{j=1}^{3} \sigma_{ij} v_j^1 = 1$, $(i = 1,2,3)$, resulting in the following system of linear equations

$$\begin{aligned} 2v_1^1 + v_2^1 &= 1 \\ v_1^1 + 2v_2^1 + v_3^1 &= 1 \\ v_2^1 + 2v_3^1 &= 1. \end{aligned}$$

The solution is $V^1 = (0.5, 0, 0.5)^T$. We next take $\alpha = 1$, $\beta = 0$ in (17.16), to solve $\sum_{j=1}^{3} \sigma_{ij} v_j^2 = \mu_i$, $(i = 1,2,3)$, i.e.

$$\begin{aligned} 2v_1^2 + v_2^2 &= 0.4 \\ v_1^2 + 2v_2^2 + v_3^2 &= 0.8 \\ v_2^2 + 2v_3^2 &= 0.4. \end{aligned}$$

The solution of the above system is $V^2 = (0.1, 0.2, 0.3)^T$.

Note that $\sum_{j=1}^{3} v_j^1 = 1$, thus we take $w^1 = V^1 = (1/2, 0, 1/2)$. Normalizing V^2, we get, $w^2 = (1/6, 1/3, 1/2)^T$ (so that, $\sum_{j=1}^{3} w_j^2 = 1$). The corresponding returns from the two portfolios with weights w^1 and w^2 are $\bar{\mu}^1 = m^T w^1 = 0.6$ and $\bar{\mu}^2 = m^T w^2 = 0.733$, respectively.

Next, an investor desired a return of $\mu = 0.334$ at minimum risk. It is simple to check that for $\lambda = 3$, $\lambda\bar{\mu}^1 + (1-\lambda)\bar{\mu}^2 = 0.334$. Thus, by the two fund theorem the requisite portfolio is given by $w = \lambda w^1 + (1-\lambda)w^2 = (7/6, -2/3, 1/2)$. Observe that the second asset has a short position in this portfolio. The variance corresponding to this portfolio is

$$w^T C w = \begin{pmatrix} 7/6 & -2/3 & 1/2 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 7/6 \\ -2/3 \\ 1/2 \end{pmatrix} = 2/9.$$

17.5 Capital Asset Pricing Model (CAPM)

So far we have assumed that all assets in the portfolio are risky assets. So it is natural to query as to what would be the scenario if one risk-free asset is included in the portfolio? In this section we make an attempt to study this aspect of portfolio selection.

Consider a portfolio with n risky assets, $a_1, \ldots, a_n$ with weights $w_1, \ldots, w_n$ and one risk-free asset a_{rf} with weight w_{rf}. Then

$$w_{\text{risky}} + w_{\text{rf}} = \sum_{i=1}^{n} w_i + w_{\text{rf}} = 1 \tag{17.17}$$

$$\Rightarrow \quad w_{\text{risky}} = \sum_{i=1}^{n} w_i \leq 1.$$

Also, the expected return and the variance associated with this portfolio are respectively given by

$$\mu = \sum_{i=1}^{n} w_i \mu_i + w_{\text{rf}} \mu_{\text{rf}} = \mu_{\text{risky}} + w_{\text{rf}} \mu_{\text{rf}},$$

$$\sigma^2 = var\left(\sum_{i=1}^{n} w_i \mu_i + w_{\text{rf}} \mu_{\text{rf}}\right) = var\left(\sum_{i=1}^{n} w_i \mu_i\right) = \sigma^2_{\text{risky}}.$$

If we remove the risk-free asset from the portfolio and readjust the weights of the risky assets so that their sum remain 1, the resultant portfolio so obtained is referred to as the *derived risky portfolio*. We use μ_{der} and σ^2_{der} to denote the derived risky portfolio expected return and risk, respectively. Then,

$$\begin{aligned}\mu &= \sum_{i=1}^{n} w_i\mu_i + w_{\text{rf}}\mu_{\text{rf}} \\ &= w_{\text{risky}}\left(\sum_{i=1}^{n}\frac{w_i}{w_{\text{risky}}}\mu_i\right) + w_{\text{rf}}\mu_{\text{rf}} \\ &= w_{\text{risky}}\mu_{\text{der}} + w_{\text{rf}}\mu_{\text{rf}}.\end{aligned} \tag{17.18}$$

$$\begin{aligned}\sigma^2 &= var\left(\sum_{i=1}^{n} w_i\mu_i\right) \\ &= w_{\text{risky}}^2 var\left(\sum_{i=1}^{n}\frac{w_i}{w_{\text{risky}}}\mu_i\right) \\ &= w_{\text{risky}}^2\,\sigma_{\text{der}}^2.\end{aligned} \tag{17.19}$$

Using (17.16) and (17.19) in (17.18), we obtain

$$\mu = \mu_{\text{risky}} + \frac{\mu_{\text{der}} - \mu_{\text{risky}}}{\sigma_{der}}\sigma\,. \tag{17.20}$$

(17.20) is an equation of a line joining $(0, \mu_{\text{risky}})$ and $(\sigma_{\text{der}}, \mu_{\text{der}})$ in the (σ, μ)-graph.

Now, for a given risk σ, suppose we choose various weight combinations of risk-free asset and risky assets satisfying (17.17), we generate different lines represented by (17.20) in (σ, μ)-graph. Obviously, among all such lines, the line that produces the points with highest expected return for a given risk is tangent to the upper portion of the Markowitz bullet. This is illustrated in Fig 17.9.

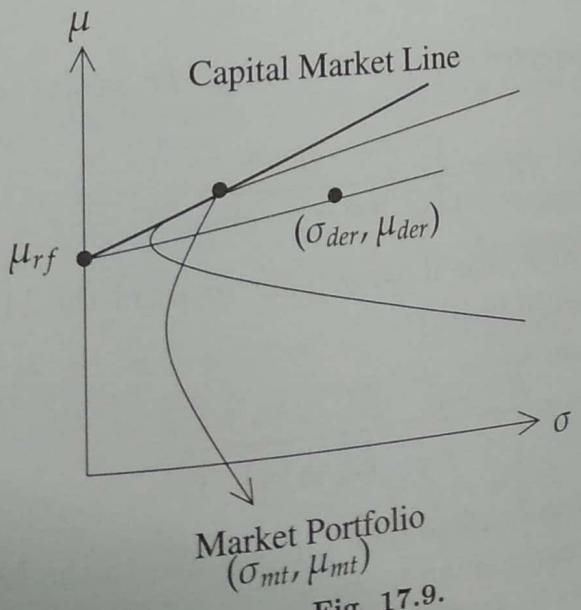


Fig. 17.9.

Definition 17.5.1 (Capital Market Line). *Among all the lines (17.20) for various weight combinations of risk-free asset and risky assets, the line giving the highest return for a given risk is called the capital market line.*

The basic idea of the capital asset pricing model (CAPM) is that an investor can improve the risk-expected return balance by investing partially in a portfolio of risky assets and partially in a risk-free asset. All investors will end up with portfolios along the capital market line as all *efficient portfolios* lie along this line while any other combination of risk-free asset and risky assets, except those which are efficient, lies below the capital market line. It is thus important to observe that all investors will hold combinations of only two assets, viz. a market portfolio and a risk-free asset. This fund scenario is summarized in the following theorem.

Theorem 17.5.1 (One Fund Theorem). *There exists a single portfolio, M, of risky assets such that any efficient portfolio can be constructed as a linear combination of the tangent portfolio and the risk-free asset.*

Unlike with the two-fund theorem where any two efficient portfolios are sufficient, in this case, the tangent portfolio is a specific portfolio.

Definition 17.5.2 (Market Potfolio). *The point on the Markowitz bullet where the capital market line is tangential is said to represent the market portfolio.*

Importance of Market Portfolio
(i) The market portfolio must contains all risky assets, for if some asset is not in it then it will wither and die.
(ii) Since the market portfolio contains all risky assets, it is a completely diversified portfolio with no unsystematic risk.

Theorem 17.5.2 *For any expected risk-free return μ_{rf}, the weights of the capital market portfolio is given by*

$$w = \frac{C^{-1}(m - \mu_{\text{rf}}\, e)}{e^T C^{-1}(m - \mu_{\text{rf}}\, e)}.$$

Proof. From Fig 17.9, we observe that for any point (σ, μ) in the Markowitz bullet, the slope of the line joining $(0, \mu_{\text{rf}})$ to (σ, μ) is

$$s = \frac{\mu - \mu_{\text{rf}}}{\sigma} = \frac{\sum_{i=1}^{n} \mu_i w_i - \mu_{\text{rf}}}{\sum_{i,j=1}^{n} c_{ij}\, w_i w_j}.$$

For the line joining $(0, \mu_{\text{rf}})$ to (σ, μ) to be a tangent line to the Markowitz bullet, we need to solve the following programming problem

$$\begin{aligned} &\text{Max} && \frac{m^T w - \mu_{\text{rf}}}{(w^T C w)^{1/2}} \\ &\text{subject to} \\ & && e^T w = 1. \end{aligned} \tag{17.21}$$

The Lagrange function $L : \mathbf{R}^n \times \mathbf{R} \to \mathbf{R}$ is described as

$$L(w, \lambda) = \frac{m^T w - \mu_{\text{rf}}}{(w^T C w)^{1/2}} + \lambda(1 - e^T w).$$

Now, solving (17.21) is same as minimizing $L(w, \lambda)$. So, $\nabla_w L(w, \lambda) = 0$, giving,

$$\frac{1}{w^T C w}\left((w^T C w)^{1/2} m - (m^T w - \mu_{\text{rf}})(w^T C w)^{-1/2} C w\right) = \lambda e.$$

The above expression can be rewritten as

$$\sigma m - (\mu - \mu_{\text{rf}})\frac{Cw}{\sigma} = \lambda \sigma^2 e.$$

Multiplying by σ, we obtain

$$\sigma^2 m - (\mu - \mu_{\text{rf}}) C w = \lambda \sigma^3 e, \tag{17.22}$$

which in turn yields

$$\sigma^2 w^T m - (\mu - \mu_{\text{rf}}) w^T C w = \lambda \sigma^3 w^T e,$$

Since $e^T w = 1$, $\mu = w^T m$, and $\sigma = w^T C w$, we get

$$\lambda = \frac{\mu_{\text{rf}}}{\sigma}. \tag{17.23}$$

The requisite value of weight vector w now follows from (17.22) and (17.23). □

Example 17.5.1 *Suppose a portfolio comprises of one risk-free asset with return* 0.5%, *and three mutually independent risky assets with expected returns* 1%, 2%, 3% *and variances* 1%, 1%, 1%, *respectively. Determine the equation of the capital market line.*

Solution The given information gives, $m^T = (\mu_1, \mu_2, \mu_3) = (1, 2, 3)$, $\mu_{\text{rf}} = 0.5$, $C = [\sigma_{ij}] = I_{3\times 3}$, $e^T = (1, 1, 1)$. Therefore, the weight vector of the market portfolio is given by

$$w_{\text{mt}} = \frac{C^{-1}(m - \mu_{\text{rf}} e)}{e^T C^{-1}(m - \mu_{\text{rf}} e)} = \begin{pmatrix} 1/9 \\ 1/3 \\ 5/9 \end{pmatrix}.$$

Consequently, the expected return and variance of the market portfolio are

$$\mu_{\text{mt}} = m^T w_{\text{mt}} = \frac{22}{9}\%, \quad \sigma_{\text{mt}} = ((w_{\text{mt}})^T C w_{\text{mt}})^{1/2} = \frac{\sqrt{35}}{9}\%.$$

Thus, the equation of the capital market line is

$$\begin{aligned} \mu &= \mu_{\text{rf}} + \frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}}\sigma \\ &= \frac{1}{2} + \frac{\sqrt{35}}{2}\sigma. \end{aligned}$$

Remark 17.5.1 *Suppose the market portfolio* $(\sigma_{\text{mt}}, \mu_{\text{mt}})$ *is known. Then, from (17.20), the equation of the capital market line is given by*

$$\mu = \mu_{\text{rf}} + \frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}}\sigma.$$

If the investor is willing to take a positive risk σ*, he can earn an additional return* $\left(\frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}}\sigma\right)$ *over and above the risk-free return* μ_{rf} *to compensate the risk taken by him.*

In practice there are certain assets which are listed in the stock called *index stocks*. These limited assets are significant ones that can capture the pulse of the whole market. The most regularly quoted market indices are broad-base indices comprising of the stocks of large companies listed on a nations' largest stock exchanges, such as the American Dow Jones Industrial Average and S&P 500 Index, the British FTSE 100, the French CAC 40, the Japanese Nikkei 225. The Bombay Stock Exchange is the largest in India, with over 6000 stocks listed and it accounts for over two thirds of the total trading volume in the country. The index stocks finally help us to compute the market portfolio $(\sigma_{\text{mt}}, \mu_{\text{mt}})$. The knowledge of the market portfolio yields the equation of capital market line, see Remark 17.5.1. Now suppose an investor P is willing to take risk σ_P. Then for this risk, the expected return μ_P is maximum if the point (σ_P, μ_P) lies on the capital market line. Thus,

$$\mu_P = \mu_{\text{rf}} + \frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}}\sigma_P.$$

If we let $w_P = \dfrac{\sigma_P}{\sigma_{\text{mt}}}$ then

$$\mu_P = w_P \mu_{\text{mt}} + (1 - w_P)\mu_{\text{rf}}.$$

Thus the expected return on an efficient portfolio can be thought of as
(Expected return) = (Price of time) + (Price of risk) × (Amount of risk).

Remark 17.5.2 *The above relation suggests that if an investor is willing to take a risk* σ_P*, then he should invest* $w_P = \dfrac{\sigma_P}{\sigma_{\text{mt}}}$ *proportion of investment in index fund and* $(1 - w_P)$ *proportion of investment in the risk-free investment schemes.*

We now aim to examine how an individual asset behaves with respect to the market portfolio. For this, we attempt to build a relationship between the expected return along with the risk of an individual asset with the market portfolio.

Theorem 17.5.3 *Suppose the market portfolio is* $(\sigma_{\text{mt}}, \mu_{\text{mt}})$. *The expected return of an asset* a_i *is given by*

$$\mu_i = \mu_{\text{rf}} + \beta_i(\mu_{\text{mt}} - \mu_{\text{rf}}), \quad \text{where} \quad \beta_{\text{i}} = \frac{\text{cov}(\mu_{\text{i}}, \mu_{\text{mt}})}{\sigma^2_{\text{mt}}}. \tag{17.24}$$

Proof. Suppose an investor portfolio comprises of asset a_i with weight w and the market portfolio M with weight $1-w$. Then the expected return and risk of the investor portfolio are respectively given by

$$\begin{aligned} \mu &= w\mu_i + (1-w)\mu_{\text{mt}}, \\ \sigma^2 &= w^2\sigma_i^2 + (1-w)^2\sigma^2_{\text{mt}} + 2\rho w(1-w)\sigma_i\sigma_{\text{mt}}, \end{aligned} \tag{17.25}$$

where ρ is the coefficient of correlation between the asset a_i and the market portfolio M.

As w varies, these values trace out a curve in the (σ, μ)-graph. It can be observed from Fig 17.10 that as w passes through zero, the capital market line becomes tangent to the curve at M. This tangency condition can be translated into the condition that the slope of the curve is equal to the slope of the capital market line at M (corresponding to $w = 0$).

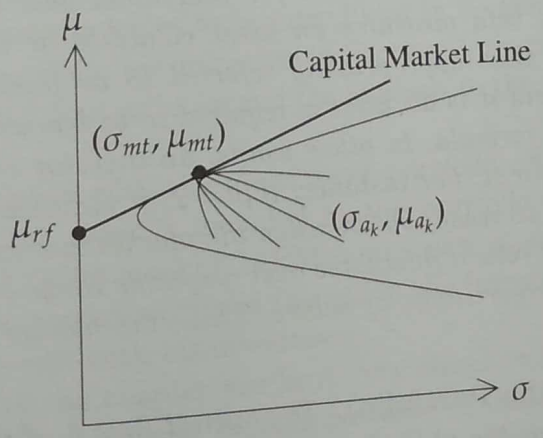


Fig. 17.10.

Now the slope of the curve at M is given by

$$\begin{aligned} \frac{d\mu}{d\sigma}(w=0) &= \frac{d\mu}{dw}\frac{dw}{d\sigma}(w=0) \\ &= (\mu_i - \mu_{\text{mt}})\frac{dw}{d\sigma}(w=0). \end{aligned}$$

Differentiating (17.25) with respect to w and computing its value at $w = 0$, we get

$$\begin{aligned}\frac{d\sigma}{dw}(w=0) &= \frac{w\sigma_i^2 - (1-w)\sigma_{\text{mt}}^2 + \rho\sigma_i\sigma_{\text{mt}}(1-2w)}{\sigma}(w=0) \\ &= \frac{\sigma_{\text{imt}} - \sigma_{\text{mt}}^2}{\sigma_{\text{mt}}}, \qquad \sigma_{\text{imt}} = \rho\sigma_i\sigma_{\text{mt}}.\end{aligned}$$

Consequently,

$$\frac{d\mu}{d\sigma}(w=0) = \frac{(\mu_i - \mu_{\text{mt}})\sigma_{\text{mt}}}{\sigma_{\text{imt}} - \sigma_{\text{mt}}^2}. \tag{17.26}$$

As discussed above, the slope of the curve needs to be equal to the slope of the capital market line at M, thereby yielding that

$$\frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}} = \frac{d\mu}{d\sigma}(w=0).$$

The above relation along with (17.26), on simplification, yields

$$\begin{aligned}\mu_i &= \mu_{\text{rf}} + \frac{\mu_{\text{mt}} - \mu_{\text{rf}}}{\sigma_{\text{mt}}^2}\sigma_{\text{imt}} \\ &= \mu_{\text{rf}} + \beta_i(\mu_{\text{mt}} - \mu_{\text{rf}}).\end{aligned}$$

□

Remark 17.5.3 *Here, $\beta_i = \dfrac{\sigma_{\text{imt}}}{\sigma_{\text{mt}}^2}$ is called the Beta of an asset. Note that, for the market portfolio, $\beta_{mt} = 1$. Beta is generally calculated for individual assets using regression analysis. As can be observed, beta measures an asset volatility or risk in relation to the rest of the market. It is thus appropriately referred to as financial elasticity or correlated relative volatility, and it is all what is required to be known about the asset's risk characteristics in CAPM formula. In other words, an investor ready to bear some systematic risk gets rewarded for it. For instance, if $\beta_i = 2$, it indicates that the ith asset return is expected to increase (decrease) by 2% when the market increases (decreases) by 1%. Equivalently, if the market return fluctuates over a specific range of values, the asset returns will fluctuate over a larger range of values. Thus, the market risk is magnified in the asset risk.*

Definition 17.5.3 (Beta of the Portfolio). *The overall Beta β of the portfolio is the weighted average of the Betas of the individual assets in the portfolio, with the weights being those that define the portfolio, i.e. $\beta = \sum_{i=1}^{n} w_i\beta_i$.*

Definition 17.5.4 (Security Market Line). *A linear equation*

$$\mu = \mu_{\text{rf}} + \beta(\mu_{\text{mt}} - \mu_{\text{rf}}), \quad \text{where} \quad \beta = \frac{\text{cov}(\mu, \mu_{\text{mt}})}{\sigma_{\text{mt}}^2},$$

that describes the expected return for all assets in the market is called the security market line. All portfolio investments lie along this line in a beta-return space. The line is pictorially depicted in Fig 17.11 in bold.

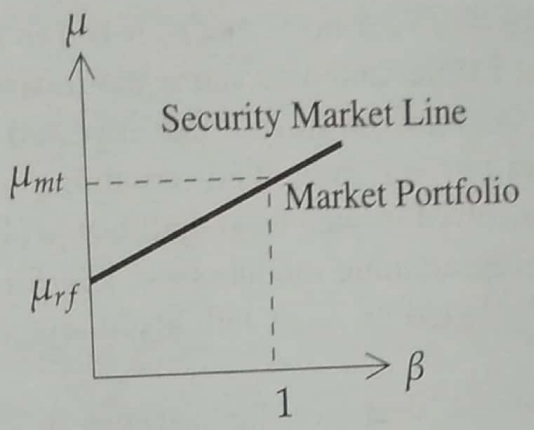


Fig. 17.11.

17.6 Summary and Additional Notes

- In this chapter we analyzed the advantages of diversifying the total investment among several assets optimally so as to get a 'decent' return with minimum risk. The theory described here is mainly based on the work of Markowitz[113].
- The requisite terminologies were introduced in Section 2, followed by a simple case of two assets portfolio optimization in Section 3, to get the feel of the subject. The ideas of Section 3 lead us to extend the analysis to the multi asset scenario in Section 4.
- Section 5 continues with the multi asset portfolio optimization with the difference that one risk-free asset is included in the portfolio. This inclusion results in the Capital Asset Pricing model (CAPM) and a new concept of market portfolio. It was shown that the market portfolio can guide the investor to determine the advantage of taking more risk with his investment.
- The contents of the chapter are kept very simple with the intention to familiarize the readers with the basics behind portfolio optimization. However things can be very complex in financial circuits. One needs to realize that any study related to financial problems requires extra skills and high level of understanding. Several interesting books covering various aspects of portfolio theory can be referred to, like, Bartholomew-Biggs [10], Capinski and Zastwawniak [31], Cornuejols and Tütüncü [39], Elton et al. [53], Luenberger [107], Roman [132], Ross [133], to name a few.
- A prominent area where portfolio optimization has gained momentum in recent years is the asset-liability management of the institutional investors, like, insurance com-

panies, pension funds or mutual funds. The institutional investors make huge investments in the markets and simultaneously repay the maturity amounts to the other investors who had invested with them. For this reasons, they need to constantly rebalance their portfolio after every time frame, which is generally very small. An institutional investor will get some inflow of money at t instance of time as the return from various investments that had been made in the market earlier and which had subsequently matured at t time, and also the institutional investor needs to pay the maturity amount to all those investors who had invested with him and whose funds have matured at the end of $t-1$ time. The remaining amount is reinvested in the market. The time scale involved in such asset-liability problems has been captured by using stochastic linear programming models (see, [88] for stochastic LPP). Number of research papers can be found on asset-liability management, like, [146, 157, 171] and references therein.

- There are several commercial packages, for instance, CPLEX, LINGO, MATLAB, SAS that provide lot of inbuilt functions for Portfolio Analysis. The major disappointment with all the commercial packages is that they can best generate only the approximate piecewise linear representation of the efficient frontier in portfolio optimization. With large number of assets involved, say 600-800, the performance of these software in computing the efficient frontier deteriorates. The MPQ (multi-parametric quadratic programming), programmed in Java and available in public domain, performs exceptionally well on large-scale applications in a reasonable time and yields the exact efficient frontier. For more on MPQ, we refer to Steuer et al. [149].
- The effect of introduction of transaction costs and/or different lending and borrowing rates in portfolio optimization theory has also been analyzed in literature. Some of the books mentioned above contain subject matter on this issue.
- Several problems of mathematical finance have been modeled as optimization problems in literature. Among them, one of the most widely studied problem is the pricing of the derivative securities, and in particular, financial options. The fundamental theorem of asset pricing showing the existence of risk neutral probability has been nicely proved using LPP duality in [39]. Besides the references listed above, one can also look for the books Ammann [3], Brigo and Mercurio [28] and David [45] for more applications of optimization problems in credit-risk models, interest rate models, volatility estimation and other financial problems.
- 'Optimal trading strategies' is another area of finance where optimization plays a key role. The mean-variance theory of Markowitz and the capital asset pricing model of Sharp tell us a great deal about what we should *hold* in our portfolio. But it is equally important to know how to *acquire* them. and this leads to the optimization of the execution cost and timing risk for obtaining an optimal trading strategy. An appropriate text on this topic is Kissell and Glantz [97]. Some important contributions in this direction are due to Almgren and Chriss [2] who introduced the concepts of

efficient trading frontiers and *capital trade line*. Bertsimas and Lo [21] studied the problem of optimal control of the execution cost by applying the stochastic dynamic programming. Recently the technique of reinforcement learning has also been used in the area of optimal trading strategies.

- Mathematical programming has also been applied to the problem of *credit scoring* where the decision models are developed that aid lenders in granting the consumer credit. These techniques assess the risk in lending to a particular consumer. An appropriate source for this topic is the text by Thomas et al. [158]

17.7 Exercises

17.1 *Suppose there are three financial market scenarios* $\Omega = \{w_1, w_2, w_3\}$ *with different probabilities of occurrence. Consider the following table showing the returns on two different stocks in these three scenarios*

scenario	prob	return k_1%	return k_2%
w_1	0.2	−10	−30
w_2	0.5	0	20
w_3	0.3	20	15

(a) What is the expected returns on the stocks?
(b) Suppose 60% *of the available fund is invested in stock 1 and the remaining is invested in stock 2, then what is the expected return of the portfolio?*
(c) Compute the weights if the expected return on a portfolio is 20%.

17.2 *Consider the following data*

scenario	prob	return k_1%	return k_2%
w_1	0.4	−10	20
w_2	0.2	0	20
w_3	0.4	20	10

Suppose a portfolio comprises of 40% *of total investment in stock 1 and* 60% *in stock 2. Compare the risk of the portfolio with the risks of its individual components. What will be the risk situation if a portfolio is designed with investment of* 80% *in stock 1 and the remaining in stock 2.*

17.3 *Prove that if short sales are not allowed then the risk of the portfolio can not exceed the greater of the risks of the individual components of the portfolio.*

17.4 *Show that if short sale is allowed in stock 1 to* 50% *and all the other data being the same as in exercise* **17.2** *the conclusion of exercise* **17.3** *fails.*

17.5 *Suppose the portfolios are constructed using three securities a_1, a_2, a_3 with expected returns, $\mu_1 = 20\%$, $\mu_2 = 13\%$, $\mu_3 = 17\%$, standard deviations of returns, $\sigma_1 = 25\%$, $\sigma_2 = 28\%$, $\sigma_3 = 20\%$, and the correlation between returns, $\rho_{12} = 0.3$, $\rho_{31} = 0.15$. Among all the attainable portfolios, find the one with minimum variance. What are the weights of the three securities in this portfolio? Also compute the expected return and standard deviation of this portfolio.*

17.6 *Among all attainable portfolios with expected return 20% constructed using the data provided in exercise* **17.5**, *find the portfolio with minimum variance. Compute the weights of individual assets in this portfolio.*

17.7 *Consider the following data*

	μ	σ
asset 1	10%	5%
asset 2	8%	2%

For each correlation coefficient $\rho = -1, -0.5, 0, 0.5, 1$, what is the combination of the two assets that yields the minimum standard deviation and what is the minimum value of the standard deviation?

17.8 *Compute the minimum risk portfolio for the following rate return (%) data:*

	Jan	Feb	Mar	Apr	May	June
asset 1	12	10	5	7	15	12
asset 2	7	12	10	10	12	15

Also compute the expected return for the optimal portfolio.

17.9 *Consider three risky assets with the covariance matrix and expected returns (all data in %) as follows.*

variance - covariance matrix(C)			return(M)
10	4	0	5
4	12	6	6
0	6	10	1

Find two portfolios yielding the minimum variance. Also, determine the expected returns from these two portfolios. Using the two fund theorem, construct the portfolio giving the return of 2.8% with minimum risk.

17.10 *Suppose an investor is interested in constructing a portfolio with one risk-free asset a_1, with risk-free return 6%, and three risky assets a_2, a_3, a_4 with expected returns 10%, 12%, 18%, respectively. Given the covariance matrix of the three assets (data in %) as*

$$C = \begin{pmatrix} 4 & 20 & 40 \\ 20 & 10 & 70 \\ 40 & 70 & 14 \end{pmatrix},$$

what is the optimum portfolio for the investor? What is the expected return of this portfolio?

17.11 *Consider the data of two risky assets* a_1, a_2 *with* $\mu_1 = 12.5\%$, $\mu_2 = 10.5\%$, $\sigma_1 = 14.9\%$, $\sigma_2 = 14\%$, $\rho = 0.33$.

(a) Is it advisable to diversify the investment? If so then what composition of the assets will minimize the risk?
(b) What is the minimum value of the risk?
(c) If the risk-free rate of return is 5% then derive the equation of the capital market line?

17.12 *Given the following information about the one risk-free asset and three risky assets, find the expected return and standard deviation of the market portfolio. Also determine the equation of the capital market line.*

$$\mu_{rf} = 5\%,\ \mu_1 = 14\%,\ \mu_2 = 8\%,\ \mu_3 = 20\%;$$
$$\sigma_1 = 6\%,\ \sigma_2 = 3\%,\ \sigma_3 = 15\%;\ \sigma_{12} = 0.5,\ \sigma_{13} = 0.2,\ \sigma_{23} = 0.4.$$

17.13 *Assume that the following assets are correctly priced according to the security market line. Derive the security market line. What is the expected return on an asset with* $\beta = 2$*?*

$$\mu_1 = 6\%,\ \beta_1 = 0.5;\quad \mu_2 = 12\%,\ \beta_2 = 1.5.$$

17.14 *If the following two assets are correctly priced according to the security market line, what is the return of the market portfolio? What is the risk-free return?*

$$\mu_1 = 9.5\%,\ \beta_1 = 0.8;\quad \mu_2 = 13.5\%,\ \beta_2 = 1.3.$$

18
Mathematical Programming Applications in Engineering

18.1 Introduction

Optimization is ubiquitous, with a multitude of applications in diverse areas. Some examples have already been listed during the course of our discussion in almost all the chapters. Besides traditional areas such as linear programming, integer programming, and mixed integer programming, some interesting areas where optimization tools have been fruitfully applied, constituted the subject matter of previous chapters on semi-definite programming, machine learning and financial mathematics.

It is always beneficial to complement the theoretical developments of the subject with some examples from real-life applications. This chapter is offered as a modest contribution in this direction. Our aim herein is to mention examples from engineering and life sciences that can be modeled as optimization problems. These problems can then be solved by theoretical tools developed in the earlier chapters.

Going with the flavor of the book, we have kept the discussion very simple by restricting ourselves to present optimization models of the problems, rather than going on to discuss their solution methodologies. In some sense this chapter can be viewed as an icing on a cake with the purpose of attracting prospective users to appreciate the power of the subject in solving interdisciplinary problems.

18.2 Optimization in VLSI Design

From its humble beginning in the early 50's to the manufacture of circuits with millions of components today, *VLSI (very large scale integration) design* has brought the power of the main frame computer to the laptop. The tremendous growth in the area of VLSI design has been made possible by the development of sophisticated tools and softwares. To deal with the complexity of millions of components, VLSI design tools must be computationally fast and generate near optimal designs.

There are various steps involved in VLSI design, like system specification (based on functionality, dimension, speed, power, choice of fabrication technique etc.), functional

design (depicting relations between various subunits), logic design, circuit design, circuit layout, design verification, fabrication (involving preparation of wafers, deposition and diffusion of various materials on the wafers), testing and debugging. Among these steps, the *circuit layout* or *geometric representation of the design* is one of the most challenging and complex processes. The circuit layout task is generally performed in two steps, viz. *a placement phase* and *a routing phase*. In the placement phase, circuits cells are assigned to locations on a layout grid. In the routing phase, inter connections between the cells are realized. We confine our discussion to the placement phase for illustrating the role of optimization in VLSI design.

The placement task seeks to assign all cells of the circuit to appropriate locations, so that cells do not overlap with each other. Each cell i is assigned a location (x_i, y_i) on the xy-plane. The cost of the wire connecting two cells i and j is proportional to the Euclidean distance between the two cells with w_{ij} being the weight. This problem leads to the following quadratic optimization formulation.

$$\text{Min} \quad \phi(x, y) = \sum_{1 \le i \le j \le N} w_{ij}((x_i - x_j)^2 + (y_i - y_j)^2)$$

subject to

$$\begin{aligned} l_x \le x_i \le u_x \\ l_y \le y_i \le u_y \ , \end{aligned} \tag{18.1}$$

where N is the number of movable cells; (x, y) are the coordinates of the movable cells; (l_x, l_y) and (u_x, u_y) are bounds on the location of the block. We note that the objective function and the constraints in (18.1) are separable in variables x_i and y_i, i.e. we can solve two independent QPPs in x and y variables.

The optimization problem (18.1) does not ensure non-overlapping placement of cells. Usually, a set of slots is defined, with each cell being required to be assigned to a separate slot. This requirement may be enforced by introducing additional constraints into the above formulation. Since the problems in the two sets of x and y variables are independent, we indicate the constraints for the problem involving only x, since the other problem can be formulated similarly.

$$\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} xslot_i. \tag{18.2}$$

This constraint is called a *linear slot constraint*. The constant $xslot_i$ indicates the x-coordinate of the i-th slot. The linear slot constraint tries to distribute the cells more uniformly. However, this constraint alone is not sufficient to ensure a non-overlapping placement of cells. In order to ensure that all cells are located in separate slots, we need to introduce more constraints, viz.

$$\sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} xslot_i^2$$
$$\sum_{i=1}^{N} x_i^3 = \sum_{i=1}^{N} xslot_i^3$$
$$\vdots$$
$$\sum_{i=1}^{N} x_i^N = \sum_{i=1}^{N} xslot_i^N . \tag{18.3}$$

However, the addition of the above described higher order slot constraints in problem (18.1) no longer guarantees convexity of the problem. Thus, the addition of these constraints would complicate the scenario considerably, thereby making the problem formulation no longer amenable to simple and efficient optimization techniques.

Instead, what is done is to first solve the placement problem (18.1) along with constraint (18.2). In the solution obtained, let the co-ordinates of the blocks be given by x_{0i}. We now solve the problem

$$\text{Min} \quad \sum_{i=1}^{N} (x_i - x_{0i})^2$$
$$\text{subject to} \quad \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} xslot_i^2.$$

This is done for different subsets of blocks; blocks outside a chosen set of blocks are kept fixed. The idea is to repeatedly solve small QPPs and arrive at an approximate solution to the original problem.

Simpler versions of placement illustrate other ways in which optimization methods play an integral role in VLSI CAD. Once again, we consider the following formulation of the placement task.

$$\text{Min} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}((x_i - x_j)^2 + (y_i - y_j)^2),$$

which may be written more compactly as

$$\text{Min} \quad \frac{1}{2} X^T B X + \frac{1}{2} Y^T B Y,$$

where $B = D - C$, with $D = diag(d_1, \ldots, d_N)$, and $d_i = \sum_{j=1}^{N} w_{ij}$ $(i = 1, \ldots, N)$.

The optimal solution of this problem is the trivial one, viz. $X = 0$, $Y = 0$. In order to have a non-trivial solution, we introduce the additional constraints

$$X^T X = 1, \quad Y^T Y = 1.$$

The readers are encouraged to write the KKT conditions for the above QPP, and show that an optimal solution must satisfy

$$BX = \lambda X, \quad BY = \mu Y,$$

implying that X and Y are the eigenvectors corresponding to the non-trivial eigenvalues of matrix B. The reader might like to think this one out: of all the eigenvalues which ones should we choose?

18.3 Global Routing for VLSI Standard Cells

In section 18.2, we discussed the *placement problem* in which layout objects are placed or located based on the interconnections between them. Placing the connected objects closer to each other helps in the next phase of the design flow - *routing*, wherein the goal is to place wires to physically realize connections. The interconnects have impact on the performance, power consumption, and the area - in fact, in current technologies, they constitute the most significant component.

Routing is usually accomplished in two phases: *a global routing phase*, which determines the regions through which connections should run, and a *detailed routing phase*. By determining which routing regions a connection runs through, global routing helps partition the detailed routing problem into smaller sub-problems that are more amenable to an efficient solution. In discussing the formulation of the problem, we follow the work of Arebti [5].

Each electrically connected set of pins is termed as a *net*. Let N_i denote the number of trees corresponding to alternate routes for net i, and let the set of possible trees for the net i be denoted by ${T_j}^i$ $(j = 1, \ldots, N_i)$. These trees can be used for routing the pins of net i. A set of variables V_j is used to indicate which tree is chosen for routing, i.e.

$$V_j = \begin{cases} 1, & {T_j}^i \text{ is chosen for routing net } i \\ 0, & \text{otherwise.} \end{cases}$$

The global routing problem is formulated as a $0-1$ LPP as follows.

$$\text{Max} \quad \sum_{j=1}^{N_i} \alpha_j V_j$$

subject to

$$\begin{aligned} \sum_{T_j^i} V_j &= 1 \\ \sum_j c_{ij} V_j &\leq C_i, \end{aligned} \tag{18.4}$$

where c_{ij} is the capacity of tree T_j^i that is used by net i, C_i is the total capacity of the i-th route tree T_j^i, and α_j is the incremental benefit obtained by wiring net i by using tree T_j^i, and it is given by

$$\alpha_j = \text{(Maximum wire length + 1) - (Wire length of tree) } T_j{}^i.$$

The first constraint in (18.4) ensures that only one tree is chosen, while the second constraint in (18.4) ensures that the capacity limit is not exhausted.

Such a problem is difficult to solve and the cost, in general, is prohibitive for large instances. Therefore, one resorts to a relaxed version of the above problem by adding the following additional constraints and removing the requirement that V_j be 0 or 1:

$$0 \leq V_j \leq 1 \quad (j = 1, \ldots, N_i). \tag{18.5}$$

Note that this leads to a linear programming problem, called the 'relaxed ILP'.

An interesting approach to solve problem (18.4) along with the constraints (18.5) is given by Raghavan [134]. The steps in Raghavan's approach are as follows.

Step 1. Solve the relaxed ILP. Let the optimal solution be given by $V = V^R$.

Step 2. Choose value for V_i by generating random numbers $\{0, 1\}$ using a biased random number generator that generates 0's with a probability of $(1 - V_i^R)$ and 1's with a probability of V_i^R.

Step 3. Repeat Step 2 until the termination criterion is met, and then go to Step 4. The termination criterion is usually based on the number of iterations of Step 2, or the improvement in the solution cost.

Step 4. Output the best feasible solution.

The logic of Step 2 stems from 'Bernoulli trials'. The expected value of V_i is V_i^R. The expected value of the objective function after several trials is given by $\sum_{j=1}^{N_i} \alpha_j V_j^R$. Thus, the expected cost of the randomized solution obtained using Raghavan's procedure is the optimal cost of the relaxed version of the problem. The described procedure can yield substantial speedups over ILP solvers, since it efficiently uses a LPP solution to efficiently obtain the solution to a much harder problem.

18.4 Wire-Sizing and Buffer Sizing

With shrinking feature sizes of VLSI devices, *interconnect delay* is increasingly becoming the dominant factor determining performance. At technology nodes below 0.25 μm or 250 nanometres (nm), the gate delay is less than the interconnect delay; in fact, in current 65 nm technologies, interconnect delay is significantly larger than gate delay. In order to reduce the delay on interconnect lines, 'buffers' are introduced in-between;

optimizing the sizes of buffers and their insertion locations has long been known to be effective in reducing delay. However, it is well known that *wire sizing* is also a very effective approach. We follow the discussion in Chu and Wong [37], and consider the wire-sizing and buffer sizing problems.

Each segment of interconnect between two buffers is modeled by using a lumped Π model, as shown in Fig 18.1.

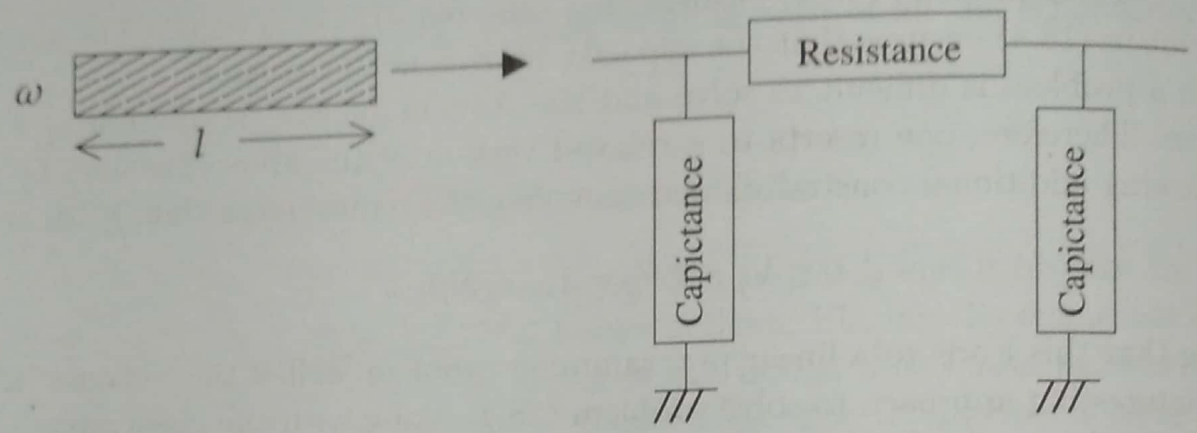


Fig. 18.1.

The capacitance of a wire segment of width ω and length l is given by $c(w)l$, where $c(w)$ is a monotonically increasing function that denotes the wire capacitance for a segment of width w and unit length.

We first consider the simple wire sizing problem. In this case, a driver (or buffer) with driver resistance R_d is driving a load capacitance of value C_L through a set of M interconnect segments of lengths $l_1, \ldots, l_M$. The task is to determine the wire length and width of each segment. The total interconnect length is specified, say, L. The wire sizing problem involves minimizing the following objective function

$$R_d\Big(\sum_{i=1}^{M} c_i l_i + C_L\Big) + \frac{\rho_s l_1}{w_1}\Big(\frac{c_1 l_1}{2} + \sum_{i=2}^{M} c_i l_i + C_L\Big) + \frac{\rho_s l_2}{w_2}\Big(\frac{c_2 l_2}{2} + \sum_{i=3}^{M} c_i l_i + C_L\Big) + \ldots + \frac{\rho_s l_M}{w_M}\Big(\frac{c_M l_M}{2} + C_L\Big)$$

$$:\equiv \frac{1}{2}L^T A L + b^T L + R_d L,$$

where $L = (l_1, \ldots, l_M)^T$, $b = (b_1, \ldots, b_M)^T$, $b_i = R_d c_i + \dfrac{C_L \rho_s}{w_i}$ $(i = 1, \ldots, M)$, and $A = [a_{ij}]_{M\times M}$, $a_{ij} = \dfrac{\rho_s c_i}{w_j}$ $(i, j = 1, \ldots, M)$.

The wire sizing task can thus be formulated as the following optimization problem

$$\text{Min} \quad \frac{1}{2}L^T AL + b^T L + R_d L$$

subject to

$$\begin{aligned} \sum_{i=1}^{M} l_i &= L \\ l_i &\geq 0 \quad (i = 1, \ldots, M). \end{aligned} \tag{18.6}$$

It can be shown that for the wire-sizing problem, the matrix A is positive definite, and hence the QPP (18.6) is a convex programming problem.

In order to reduce the delay along a long interconnect, it is usually divided into segments and buffers are inserted after each segment. The sizes of the buffers $B_1, \ldots, B_N$ need to be chosen for an optimal delay, since the last buffer drives the load capacitance (usually large), while the first one drives only the capacitance of an interconnect segment. In general, optimization strategies yield results in which the buffer sizes increase from the first to the last, with each buffer driving a progressively larger load.

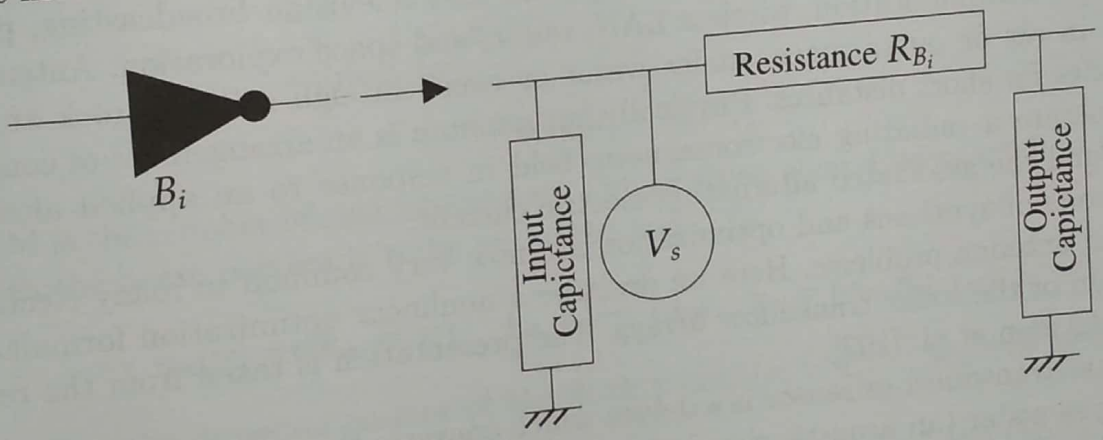


Fig. 18.2.

Fig 18.2 shows a sketch of a typical buffer and a simple circuit model for the same. Without loss of generality, we can consider the situation as that of a driver (the buffer of size B_i) driving a load of size B_{i+1}. The wire-sizing problem can be integrated with this task by dividing the interconnect between any two buffers into M segments, and then solving the wire-sizing QPP(18.6).

We observe, that with this model, the circuit model is similar to the earlier one used to model connected segments of an interconnect, the only difference being that the resistance and capacitance values in the model depend on the buffer size rather than the interconnect size.

The buffer sizing problem thus leads to a quadratic programming problem of the form

$$\text{Min} \qquad \frac{1}{2}L^T DL + g^T L + \sum_{k=0}^{N} R_{bk} C_{B_{k+1}} + \sum_{k=0}^{N} \tau_{B_k}$$

subject to

$$\begin{aligned} \sum_{i=1}^{M(N+1)} l_i &= L \\ l_i &\geq 0 \quad (i = 1, \ldots, M). \end{aligned}$$

18.5 Synthesis of Antennae Array

An important engineering application of optimization problems is the *synthesis of arrays of antennae.*

An antenna is an electromagnetic device that can generate and receive electromagnetic waves. Antennas convert electromagnetic waves into electrical currents and vice versa. They are used in systems such as radio and television broadcasting, point-to-point radio communication, wireless LAN, radar, and space exploration. Antennas can operate in air or outer space, under water or even through soil and rock at certain frequencies for short distances. Physically, an *antenna* is an arrangement of conductors that generate a radiating electromagnetic field in response to an applied alternating voltage and the associated alternating electric current.

Methods of synthesis and optimization are now very common to many electromagnetic and antenna problems. Here we describe a nonlinear optimization formulation of the design of the sonar transducer arrays. The presentation is taken from the research work of Lasdon et al. [102].

A sonar transducer or sensor is a device which converts the energy of the sound wave traveling in water (an acoustic signal) into an electric signal.

Consider a sensor whose position is specified by the vector $r = (x, y, z)$. Acoustic plane waves of wavelength λ and frequency f, incident in a direction specified by the unit vector $u = (\alpha, \beta, \gamma)$, impinge upon this sensor. Using spherical coordinates, we have

$$\cos\alpha = \sin\phi\cos\theta, \quad \cos\beta = \sin\phi\sin\theta, \quad \cos\gamma = \cos\phi.$$

Another quantity called 'wave number', is defined as $k = \left(\frac{2\pi}{\lambda}\right)u$. If the incoming signals are assumed to have unit amplitude and zero phase at the origin, then the sensor output *os* is described by a complex number as follows.

$$os = B(k)\exp(ik^T r),$$

where $B(\cdot)$ is the *response function* denoting the response of the sensor to waves of different wavelengths incident from different directions. Also, $k^T r$ is the phase term arising

because the plane wave reaches the sensor $\frac{r^T u}{f\lambda}$ seconds before it reaches origin. Associated with each sensor are two adjustable parameters - a complex shading coefficient w and a steering phase $r^T ks$. After these operations are applied, the sensor response rs is given by

$$rs = w\,B(k)\,\exp(i r^T(k - ks)).$$

Let there be a linear array of N acoustic antennae placed at fixed positions r_j $(j = 1\ldots,N)$. The N array response, called the 'beam pattern', is obtained by summing over all sensors, i.e.

$$a(k, ks, w) = \sum_{j=1}^{N} w_j\,B_j(k)\,\exp(i r_j^T(k - ks)).$$

Suppose we wish to minimize the beam pattern level over a given zone with the possibility of level constraints in other areas. The problem can be formulated as the following optimization problem.

$$\begin{aligned} &\underset{w}{Min} \quad \underset{1 \le i \le M}{Max} \quad |a_m(w)| \\ &\text{subject to} \\ &\qquad a_0(w) \quad = 1, \end{aligned} \tag{18.7}$$

where M is the number of side lobe regions (whose levels should be as small as possible) at which the beam pattern is to be evaluated and

$$a_m(w) = a(k_m, ks, w), \quad k_m = \left(\frac{2\pi}{\lambda}\right) u_m, \quad u_m = (\alpha_m, \beta_m, \gamma_m).$$

u_m specified the direction cosines of the m-th point, $(m = 1,\ldots,M)$. For optimization purpose the steering direction ks is fixed.

The complex functions and complex variables appearing in problem (18.7), when expressed in terms of the real functions and real variables, leads to the following nonlinear constrained optimization problem.

$$\begin{aligned} &\text{Min} \quad z \\ &\text{subject to} \\ &\qquad (a_m^T x)^2 + (b_m^T x)^2 \quad \le z, \quad (m = 1,\ldots,M) \\ &\qquad a_0^T x \quad = 1 \\ &\qquad b_0^T x \quad = 0 \\ &\Big(\sum_{k=1}^{N} x_k\Big)^2 + \Big(\sum_{k=1}^{N} x_{n+k}\Big)^2 - \quad \overline{N}_e \sum_{k=1}^{N} (x_k^2 + x_{n+k}^2) \ge 0. \end{aligned} \tag{18.8}$$

The quantity $\overline{N}_e$ is the effective number of elements. Problem (18.8) may be solved by the Lagrangian relaxation algorithm. For more details on optimization of antennae design, we refer the reader to [16].

18.6 Chemical Equilibrium

We begin with a simple example pertaining to *chemical equilibrium*. The example is taken from White et al. [164]. Chemical equilibrium is the state in which the chemical activities or concentrations of the reactants and products experience no net change over a period of time. Usually, this would be the state that results when the forward chemical process proceeds at the same rate as its reverse reaction.

The chemical equilibrium problem requires us to determine the chemical composition of a complex mixture, containing m different types of chemical elements, at chemical equilibrium. The governing principle behind chemical equilibrium is the 'second law of thermodynamics'. The thermodynamic condition for chemical equilibrium is, that a chemical mixture held at a constant temperature and constant pressure reaches its chemical equilibrium state when the *Gibbs free energy* of the mixture is minimum. The problem therefore involves minimizing the Gibbs free energy of the mixture subject to chemical reactions possible between the chemical elements of the mixture.

Before describing the mathematical model of the problem, we explain the notation that will be used in the problem formulation. Let

m be the total number of chemicals elements in the mixture;

n be the total number of chemical compounds formed after the chemical reactions;

x_i be the number of moles of compound i present in the mixture at equilibrium;

a_{ij} be the number of atoms of element i in a molecule of compound j;

b_i be the number of atomic weights of element i in the mixture;

$\hat{x} = \sum_{i=1}^{n} x_i$;

P be the total atmospheric pressure, R be the gas constant, T be the absolute temperature, and F be the Faraday constant.

White et al. [164] formulated the chemical equilibrium problem as follows.

$$\text{Min} \quad \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} x_i \ln\left(\frac{x_i}{\hat{x}}\right)$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, \ldots, m)$$

$$x_j \geq 0 \quad (j = 1, \ldots, n),$$

where $c_i = \left(\frac{F}{RT}\right)_i + \ln(P), \quad (i = 1, \ldots, n).$

The above problem is a nonlinear programming problem with linear and nonnegativity constraints.

18.7 Protein Folding

We now cite an example from biological sciences that is of great interest to applied optimization researchers. One of the most important and difficult problem in molecular biophysics and biochemistry is the *protein folding problem.* The protein folding problem is simply stated as:

Given a known primary sequence of amino acids, predict its native or folded state in 3-dimensional space, i.e. predict how newly made proteins, which resemble loosely coiled strands and are inactive in their unfolded configurations, will *fold* into specifically shaped balls able to perform crucial tasks in a living cell.

Let the molecules to be folded consist of a linear sequence of n beads $a_1, \ldots, a_n$, where a_i denotes the i-th bead in the primary sequence. Let l_i be the length of the i-th bond, i.e. the distance between two nuclei of atoms of consecutive beads a_i and a_{i+1}. For example, the equilibrium bond length of Carbon-Nitrogen (C-N) is 1.335 Å. For three consecutive beads a_{i-1}, a_i, a_{i+1}, let θ_i represent the 'bond angle' corresponding to the position of the third bead a_{i+1} with respect to the line joining a_{i-1} and a_i. For example, the equilibrium bond angle of Carbon-Nitrogen-Hydrogen (C-N-H) is 118.8°, while for Oxygen-Carbon-Nitrogen (O-C-N) it is 122.9°. A bond angle around 109° means that the central atom is tetrahedral. For four consecutive beads, a_{i-2}, a_{i-1}, a_i, and a_{i+1}, let ϕ_i denote the 'torsion angle' or 'dihedral angle' between the two planes described by a_{i-2}, a_{i-1}, a_i and a_{i-1}, a_i, a_{i+1}. This is illustrated in Fig 18.3.

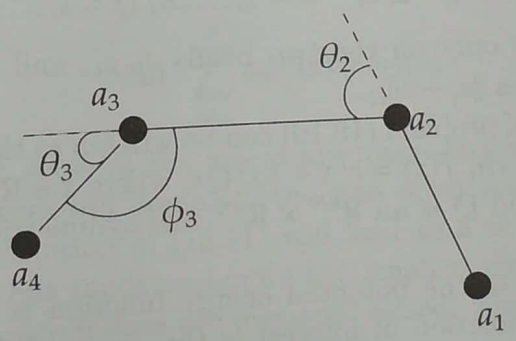


Fig. 18.3.

Let $y = ((l_1, \theta_1, \phi_1), \ldots, (l_{n-1}, \theta_{n-1}, \phi_{n-1})) \in \mathbf{R}^{3n-3}$, with $\theta_1 = 0$, $\phi_1 = 0$, $\phi_2 = 0$, and let $f(y)$ be an appropriate potential energy function. Then, the problem of molecular conformation involves determining the global min point of the problem

$$Min \quad f(y). \tag{18.9}$$

Because of the large number of variables y, it is hard to determine the global optimum solution of (18.9). Instead of solving problem (18.9) directly, Phillips and Rosen [123],

proposed to solve it in two stages. We describe only the first step of the proposed scheme in order to give readers feel for optimization problem; details may be found in [123].

In the first step, problem (18.9) is approximated by a discrete problem. The 3-dimensional space is approximated by a 3-dimensional lattice with N cites, $N \geq n$. Let these sites be represented by s_j $(j = 1, \ldots, N)$. Introduce variables x_{ij} as follows.

$$x_{ij} = \begin{cases} 1, & \text{if } a_i \text{ is assigned to } s_j \\ 0, & \text{otherwise.} \end{cases}$$

Only two types of constraints are prescribed: (i) each bead must occupy exactly one lattice site; (ii) at most one bead occupies each lattice site s_j.

The first stage problem is thus formulated as the following quadratic assignment problem.

$$\text{Min} \quad \sum_{i=1}^{n}\sum_{j=1}^{N} d_{ij}\, x_{ij} + \frac{1}{2}\sum_{1=1}^{n}\sum_{j=1}^{N}\sum_{k=1}^{n}\sum_{l=1}^{N} p_{ijkl} x_{ij} x_{kl}$$

subject to

$$\begin{aligned} \sum_{j=1}^{N} x_{ij} &= 1, \quad (i = 1, \ldots, n) \\ \sum_{i=1}^{n} x_{ij} &\leq 1, \quad (j = 1, \ldots, N) \\ x_{ij} &\geq 0, \quad (i = 1, \ldots, n)\ (j = 1, \ldots, N). \end{aligned} \tag{18.10}$$

The term p_{ijkl} depends only on the two beads a_l, a_k, and the Euclidean distance between their allocated sites $\|s_j - s_l\|_2$.

The objective function of problem (18.10) can be written as the total potential energy function in the quadratic form $f(x) = c^T x + \frac{1}{2} x^T Q x$, where $x \in \mathbf{R}^{nN}$ is a zero-one vector, $c \in \mathbf{R}^{nN}$ with entries d_{ij}, and Q is an $\mathbf{R}^{nN} \times \mathbf{R}^{nN}$ real symmetric matrix. The diagonal elements of Q are zero.

The appropriate choice of the potential energy function is a crucial factor in the entire analysis. One such function of interest is the *Lennard-Jones pairwise potential* between beads a_i and a_k, and is defined as

$$f_{ik}(r) = \epsilon_{ik}\left(\left(\frac{\sigma_{ik}}{r}\right)^{12} - 2\left(\frac{\sigma_{ik}}{r}\right)^{6}\right),$$

where ϵ_{ik} and σ_{ik} are constants related to the two specific beads a_i and a_k. This function is used to describe the quadratic term,

$$p_{ijkl} = f_{ik}(r_{jl}),$$

where r_{jl} is the distance between the lattice sites s_j and s_l. The above discussion when summarized shows that solving the quadratic assignment problem (18.10) is equivalent to solving an unconstrained quadratic 0-1 program.

18.8 Curve Fitting

We now describe a problem which we encounter in nearly every branch of engineering, ranging from biology to computer science, geometry to electric engineering, and from finance to pattern classification. The problem is that of *curve fitting*. With increasing interest in data analysis in recent years, the problem and its advanced versions have become increasingly important. As usual, we restrict ourselves to describing the fundamental problem and its optimization model, without looking into its advance aspects.

Consider the problem of fitting a model of the form $y = h(v,x)$ to a set of observed data. Here, vector v denotes an independent variable and vector y represents the dependent variable. We are required to determine the vector x of model parameters such that the model $h(v,x)$ matches the set of observed data as closely as possible.

Suppose there are p sets of observations, say, (y_i, v_i), $(i = 1,\ldots,p)$. We have to find x such that the deviation in the predicted value of the dependent variable y_i for a given value v_i by the model, from the actual observed value y_i is a minimum. There are various rules to measure this error. The easiest and perhaps the most natural way is to minimize the sum of squares of the errors e_i, where

$$e_i(x) = y_i - h(v_i,x) \quad (i = 1,\ldots,p).$$

In certain problems it is desirable to weight some errors more heavily than the others. This is achieved by associating non-negative weights w_i, with the error variables, leading to the problem

$$\text{Min} \sum_{i=1}^{p} w_i e_i^2(x) = n_2(x). \tag{18.11}$$

It may happens that some of the errors are much larger than others. The sum of squares of the larger errors will dominate the sum of squares of the smaller errors. In such case, the optimal solution of (18.11) will lead to a fit $h(x,v)$ that tries to make the larger errors small but tends to ignore the smaller errors. In other words, some outlier entries in the observed data are overemphasized. This ambiguity is removed by considering the following optimization problem.

$$\text{Min} \sum_{i=1}^{p} w_i |\, e_i(x)\,| = n_1(x). \tag{18.12}$$

Observe that problem (18.12) is not differentiable at points where one or more errors $e_i(x) = 0$. However, it can be converted into the following constrained differentiable optimization program.

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{p} w_i(\xi_i + \eta_i) \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} e_i(x) &= \xi_i - \eta_i \quad (i = 1, \ldots, p) \\ \xi_i,\ \eta_i &\geq 0 \qquad (i = 1, \ldots, p). \end{aligned}$$

There are other ways of considering the error. In some problems, it is important to focus only on the largest errors while ignoring the others. In other words, for an arbitrary vector x, determine the maximum absolute error, $\underset{i}{Max}\ |e_i(x)|$. Thereafter, find a vector x that solves the following optimization problem.

$$\underset{x}{Min}\ \underset{i}{Max}\ |e_i(x)|. \tag{18.13}$$

Such optimization models are frequently encountered in sophisticated computer aided design (CAD) problems. Mechanical CAD is used to design surfaces and shapes of 3-dimensional bodies. A smooth, mathematically defined surface is fit to the given set of data points in 3D. The surface is often chosen to be a typical polynomial of low degree, called a *spline*. The coefficients of the surface polynomial are determined by solving an optimization problem (18.13).

We observe that problem (18.13) is not differentiable at points x where two or more errors $e_i(x)$ are equal. It is easy to see that problem (18.13) is equivalent to the following smooth nonlinear optimization problem

$$\begin{aligned} \text{Min} \quad & z \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} e_i(x) &\leq z \quad (i = 1, \ldots, p) \\ e_i(x) &\geq -z \quad (i = 1, \ldots, p) \\ z &\geq 0. \end{aligned}$$

Remember, $e_i(x) = y_i - h(x, v_i), \quad (i = 1, \ldots, p)$.

18.9 Reliability Optimization

One of the natural tasks an engineer is often faced with during the process of developing a new product, is to design a system such that it conforms to a set of reliability specifications. The probability that a component survives until some specified time is called the *reliability of the component*. It is obvious that the reliability of the entire system depends on the reliability of several components constituting that system. The aim is to work out a system design that will meet the desired reliability specification while performing all of the intended functions at a minimum cost.

Reliability optimization is a subject that caters to this aspect of engineering design. Reliability optimization is an integral part of the design and operation of large scale industrial systems. One needs to answer questions such as 'How to design a complex system, such as a spacecraft, a modern day computing system, etc., to achieve the longest life cycle at the lowest cost?' The solution involves a balancing act of allocating reliability to the components in the system so the system will meet its reliability goal while at the same time ensuring that the system meets all the other associated performance specifications.

The reliability optimization process begins with the development of a model that can represents the entire system. This task is accomplished by constructing a system reliability block diagram that represents reliability relationships of the components of the system. In order to illustrate the formulation of optimization problems in reliability theory, we cite a simple example from Shelokar et al. [142].

Consider the system design shown in Fig 18.4.

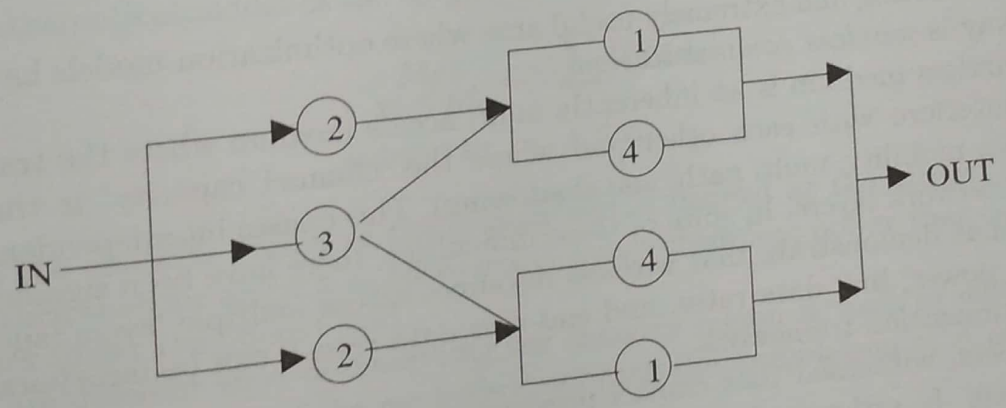


Fig. 18.4.

Here, unit 1 is backed up by a parallel unit 4; there are two equal parallel paths, each of which has unit 2 in series with a stage formed by unit 1 and unit 4; unit 3 is inserted in the system to enhance its reliability.

The following operations are possible: 2 – 1, 2 – 4, 3 – 1, 3 – 4, and each operation has equal paths. The overall system reliability is given by (1 - system failure), i.e.

$$R_S = 1 - R_3((1 - R_1)(1 - R_4))^2 - (1 - R_3)(1 - R_2(1 - (1 - R_1)(1 - R_4)))^2.$$

The problem is to maximize the system reliability so that the total approximate cost would be minimum. The two-objective nonlinear optimization model is as follows:

$$\begin{aligned}
&\text{Max} \quad R_S = 1 - R_3((1-R_1)(1-R_4))^2 - (1-R_3)(1-R_2(1-(1-R_1)(1-R_4)))^2 \\
&\text{Min} \quad C_S = 2K_1R_1^{\alpha_1} + 2K_2R_2^{\alpha_2} + K_3R_3^{\alpha_3} + 2K_4R_4^{\alpha_4} \\
&\text{subject to} \\
&\qquad R_i \geq 0.5 \quad (i = 1, \ldots, 4) \\
&\qquad R_S \geq 0.9 \\
&\qquad C_S \leq 700.
\end{aligned}$$

Here, $K_1 = 100$, $K_2 = 100$, $K_3 = 200$, $K_4 = 150$, $\alpha_i = 0.6$ $(i = 1, \ldots, 4)$.

The above example is a small illustration of the role of optimization in reliability theory. The subject is vast and one often encounters multiobjective optimization problems set up in different settings while formulating reliability models. An interested reader may like to refer to [98].

18.10 Wireless Network System

Another interesting and extremely useful area where optimization models have come up in a big way is *wireless communication.*

The wireless medium is an inherently multi access medium where the transmissions of users interfere with each other and where the 'channel capacity' is time varying (due to user mobility, multi path, and shadowing). This causes interdependencies across users and network layers. In spite of these difficulties, there have been significant recent advances that demonstrate that wireless resources across multiple layers (such as time, frequency, power, link data rates, and end-user data rates), can be incorporated into a unified optimization framework.

It has been witnessed that convex programming is an important tool for this optimization task. In order to deal with complex optimization problems, Lagrange duality is extensively used as a key tool in decomposing the problem into easily solvable components. At the same time, one needs to realize that convexity is often not enough to describe the system completely. The essential features of many wireless cross-layer control problems are non-convex.

We now present a simple problem to illustrate optimization-based approach for resource allocation problems in wireless systems.

Consider a multihop wireless network with N nodes. Let $\mathcal{L}$ denote the set of node pairs (i, j) such that transmission from node i to node j is allowed. The data rate r_{ij} of a link depends on power P_{ij} assigned to the link, and also on the interference due to the power assignments on other links.

Let $P = \{P_{ij} \,|\, (i,j) \in \mathcal{L}\}$ denote the power assignment and $r = \{r_{ij} \,|\, (i,j) \in \mathcal{L}\}$ denote the data rate. We assume that the data rate is completely determined by a global power assignment, i.e.

$$r = u(P).$$

The function $u(\cdot)$ is called the 'rate-power function' of the system.

Suppose there are $\mathcal{S}$ users and each user is associated with a source node f_s and a destination node d_s. Let x_s be the rate at which data is sent from f_s to d_s, possibly over multiple paths and multiple hops. It is natural to assume that x_s is bounded by $[0, M_s]$. Each user is associated with a utility function, which reflects the *utility* to the user when it transmits data at rate x_s. The utility function $U_s(\cdot)$ is generally taken to be strictly concave, nondecreasing and continuously differentiable on $[0, M_s]$. The joint congestion-control and scheduling problems are formulated as follows.

The congestion-control problem. Find the user rate vector x that maximizes the sum of the utilities of all users subject to the constraint that the system is stable under some scheduling policy.

The scheduling problem. For any user rate vector x picked by the congestion-control problem, find a scheduling policy that stabilizes the system.

A system is said to be *stable* under a scheduling policy if the queue length at each node remains finite.

The mathematical model of the congestion-control problem is as follows.

$$\begin{array}{ll} \underset{x_s \le M_s}{Max} & \sum_s U_s(x_s) \\ \text{subject to} & x \in \Delta. \end{array}$$

Here Δ is the capacity region of the system and is defined as the largest set of rate vectors x such that for any $x \in \Delta$, there exists some scheduling policy that can stabilize the network under the offered load.

There are different ways of describing the capacity region Δ, each of which leads to a different solution. Some of the methods describing the formulation of Δ are reviewed in [103].

18.11 NMR Experiment Design

We now move to the area of *Nuclear Magnetic Resonance (NMR) experiments* to illustrate an application of semi-definite programming. NMR experiments lead to model fitting problems, where the model function is governed by the physical properties of the NMR signals. Designing NMR experiments is a creative and challenging process that requires both physical and mathematical insight into the experiment.

Due to the low signal-to-noise ratio, a large amount of data is collected in order to estimate the experimental parameters accurately. This results in a large acquisition time. We present the problem of optimally choosing the experimental variables in order to minimize the uncertainty in the estimates obtained from NMR experiments. We will show how semi-definite optimization problems arise as a natural consequence of the optimality criteria.

The following are the most common features of the standard design problem.

(i) The vector of variables x known as experiment variables that can be controlled.
(ii) The unknown parameters $\theta \in \mathbf{R}^m$ known as experimental parameters which need to be estimated from the observe responses.
(iii) The observed responses denoted by y.

The quantitative model relating the above described variables is given by

$$y_i = \eta(x_i, \theta) + \epsilon_i \quad (i = 1, \ldots, n),$$

where ϵ_i, $(i = 1, \ldots, n)$ are uncorrelated normally distributed random noise sources with zero mean and variance σ^2.

We begin the discussion by assuming that the 'response function' $\eta(x, \theta)$ is a linear function of θ, i.e.

$$\eta(x, \theta) = \theta^T f(x).$$

Here $f(x)$ is termed as the vector of 'basis functions' and is assumed to be known. The 'least squares unbiased estimator' $\hat{\theta}$ of θ is defined as

$$\hat{\theta} = \min_{\theta} \sum_{i=1}^{n} (y_i - \theta^T f(x_i))^2.$$

Then,

$$Var(\hat{\theta}) = \frac{\sigma^2}{n} \mathfrak{I}^{-1}(x)$$

$$\mathfrak{I}(x) = \sum_{i=1}^{n} f(x_i) f(x_i)^T.$$

Here $\mathfrak{I}(x)$ is the *Fisher information matrix*. The Fisher information matrix is a way of measuring the amount of information that an observable random variable x carries about an unknown parameter θ, upon which the likelihood function of θ depends. Note that the Fisher information matrix is a symmetric and positive semi-definite.

The likelihood function is the joint probability of the data x, conditional on the value of θ, as a function of θ. Since the expectation of the score is zero, the variance is simply the second moment of the score, the derivative of the log of the likelihood function with respect to θ, i.e.

$$E\left\{\left[\frac{\partial}{\partial \theta} \ln \eta(x, \theta)\right]^2 | \theta\right\}.$$

The Fisher information is thus the expectation of the squared score. It is named in honor of its inventor, the English statistician R. A. Fisher.

Based on information matrix theory, various types of optimality criteria have been adopted by design specialists. Several of these criteria are based on functions of eigenvalues of the inverse of the information matrix. One such optimality criterion is known

as *E-optimality*, which involves minimizing the maximum eigenvalue of the inverse information matrix. Using the notion of the Fourier transform [145], the optimization problem has been re-casted as a semi-definite programming problem in [8]. We skip the model formulation for the sake of brevity; our basic aim has been to highlight the breadth of applications of optimization theory in real world problems. The interested readers may refer to [8].

18.12 Summary and Additional Notes

- A great part of the engineering design process is and will always be intuitive, however analytical techniques as well as numerical optimization could be of great value and can permit vast improvements in design. Engineering design clearly requires not only incorporating physical constraints impacting the design, but also in-depth mathematical aspects of the model. The latter ultimately helps us to find appropriate solution methodologies for the problems.
- In this chapter, we attempted to illustrate the breadth of applications of optimization theory in engineering design. Some simple problems have been taken from electric engineering, chemical engineering, biological sciences, reliability theory, and communication. Although these problems are not exhaustive by any means, they provide the readers with a good sample of the diversity of optimization applications. The chapter provides only a sketch of various problems, and intentionally avoids any serious computational aspect. One can take a look at [46, 50, 79, 127] for more applications of optimization in other engineering sciences.
- In this chapter, we targeted the engineering aspects of optimization only, but it is a well established fact that optimization problems are frequently encountered in several areas of management, including supply chain management, facility location problems, water management, agriculture management, health sector, scheduling and planning problems, replacement theory, and network designs, to name a few. Books by Hiller and Liebermann [76], Rardin [135], Taha [154], and Winston [165] include numerous examples illustrating importance of optimization in management.

19
Matlab Codes For Some Selected Algorithms

19.1 Introduction

MATLAB, which stands for MATrix LABoratory, is a state of the art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of sciences and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation.

MATLAB deals primarily with numbers and collections of numbers called "vectors" and "matrices." The concept of variable and value is very similar to other programming languages. A variable is a container that holds a value.

Create a variable and assign it a value

In MATLAB, the value is typically a number or a vector or a matrix of numbers. Each variable has a name, such as var1. There are four things one does with variables: create them, assign them a value, change the assigned value, and use the value that has already been assigned.

The following command creates a variable named a and assigns it the value 7. To execute the command, type the command at the prompt (>>) and press Enter. (Note: no need to type the prompt, the computer produces this itself.)

$>> a = 7$

$ans : a = 7$

Using a variable that has already been defined

To see what is the value of a variable, just type its name at the command prompt, e.g. $>> a$.

To use the value of a variable in a calculation, just type the name of the vari-

able where ever one would like the value to appear. For instance, >> $a + a$ adds a to itself. Arithmetic operations follow the conventional notation. Try the following >> $a = 2,\ b = 3,\ c = (a + b)^2$. Note that ^{2} means exponentiation to the power 2.
>> $d = (a - b)/(a + b)$
After each assignment, MATLAB prints out the value of variable. This can sometimes be irritating. One can use a semi-colon at the end of a line in order to suppress this printing. Try:
>> $e = 2 * a + 7 * b;$

Use of Function
Whenever a function takes arguments, the arguments are enclosed in a pair of parentheses. If there are more than one argument, they are separated by commas, and the order of the arguments is usually extremely important. One can find out what a function does, and what its arguments mean by giving the command: help name of function.
Try >> help rand.
In most of the cases, the result of using a function is a value. For example, the value of sqrt(25) is 5. One can use this value in the similar way as one would have used any other value. For example:
>> $b = cos(a)$
or
>> $c = 3.5 * cos(a * b) + sin(sqrt(a))$

Handling of Vectors and Matrices

MATLAB deals with collections of numbers called vectors and matrices. A vector is simply a list of numbers. A matrix is a rectangular array of numbers. There are many ways to make a vector; one of the most useful ways is to make a sequence using the colon (:) sequencing operator.
Try the following commands:
>> $a = 1 : 10;$
>> $b = 5 : 10;$
>> $c = 9 : 0.5 : 11.2$ which translates as assign c to be the sequence from 9 to 11.2, stepping by 0.5. Note that 11.2 is not in the sequence since one can't get to 11.2 by taking steps of size 0.5 starting from 9.
To enter matrix try following command
>> $mat = [12; 34; 56; 78]$.
Note that the rows of the matrix are separated by the semi-colon (;). All rows of a matrix must be the same length. The printed form of the matrix reveals the rectangular structure clearly.
Other common operations on vectors
Some operations on vectors give back a single number, or perform some other operation

such as plotting the vector.
Try
>> $d = 1 : 10$
>> $mean(d)$ average of the numbers in d.
>> $std(d)$ standard deviation of the number in d.
>> $median(d)$
>> $plot(d)$
>> $sum(d)$
>> $plot(d, sin(d))$

Function Handling

A function handle is a MATLAB value that provides a means of calling a function indirectly. One can pass function handles in calls to other functions (often called function functions).

handle = @functionname returns a handle to the specified MATLAB function.

Vector Handling

Some MATLAB functions support only vector inputs, others accept matrices.

When the data is a vector, the result is the same whether the vector has a row wise or column wise orientation.

However, when the data is a matrix, MATLAB performs calculations independently for each column. This means that when one passes a matrix as an argument to the function max, for example, the result is a row vector containing maximum data values for each column in the matrix.

Note: When the data is a matrix where each row contains a data set, one must transpose the matrix before proceeding with the data-analysis tasks to make the data sets have a column wise orientation. For example, to transpose a real matrix A, use the syntax A'.

Another important operation that MATLAB can perform with ease is matrix division. If M is an invertible square matrix and b is a compatible vector then $x = M\backslash b$ is the solution of $Mx = b$ and $x = b/M$ is the solution of $xM = b$.

We next present MATLAB codes for some of the algorithms discussed in this book. For writing codes for nonlinear optimization problems, we have restricted ourselves to the *quadratic case* only. This has been done for better understanding of these codes. These codes can certainly be modified to the general nonlinear case as well. But this will require appropriate modifications in inputting the data by employing symbolic functions of matlab.

Note: The readers are encouraged to use the "help" facility of MATLAB to understand more about MATLAB itself and also to tackle difficulties faced while coding.

19.2 Simplex Algorithm

```
% The simplex algorithm consist of four basic steps. These are (1)
% printing simplex tableau (2) identifying column to enter (3)
% identifying column to leave, and (4) pivoting. We introduce
% each of these functions here. First we discuss the case when
% artificial variables are not required and next when
%  artificial variable are required.

function [E] = simplextableau(Y,Z,Xb,U,V,fval)

% This function displays and prints the simplex tableau in
% the condensed form. Here, the first row corresponds to
% the indices of nonbasic variables and the last column
% corresponds to the indices of basic variables. Last row
% corresponds to the values of Z_j-c_j, and the first column
% corresponds to values of basic variables.
% Y is the set of columns of the simplex tableau which correspond
% to non basic variables
% fval is the value of the objective function corresponding
% to the current bfs Xb.
% U (V) is a vector of indices corresponding to basic (non basic)
% variables
% Z is a vector whose entries correspond to the values of zj-cj

[m,n]=size(Y);
[M,N]=size(V);
E=zeros(m+2,N+2);% creates vector of zeros
for i=1:N
    E(2:(m+1),i+1)=Y(:,V(i));
     %assigns column corresponding to non basic variables from Y to E
    E(m+2,i+1)=Z(V(i));
end
E(1,2:(N+1))=V;
E(2:(m+1),N+2)=U';
E(2:(m+1),1)=Xb;
E(m+2,1)=fval;
return;
```

```
function [z2] = findDepartindex(E,z1,M,N)
% This function identifies the departing variable index

%z1 is the index of the entering column

k=1;
for i=2:(M-1)% do for all bv's
    if(E(i,z1)>0)
        ratio(k) = E(i,1)/E(i,z1);% min ratio if y_ij is >0
    else
        ratio(k)=Inf;%else inf
    end
    k=k+1;
end

[r]=find(ratio==min(ratio));% find the index corr to min ratio
r=max(r);

z2=r(1)+1; %z2 corr to index of departing row
return;

% This function removes a column from matrix

function [B]=remove(A,col)
%col is the index of column which to be deleted from matrix A
[m,n]=size(A);

B=A(:,1:(col-1));% extracting column from 1: (col-1) from A

B=[B A(:,(col+1):n)]; %adding column from (col+1):n of A

return;
```

```
% This function updates the simplex tableau using pivoting

function [E] = pivoting(E,z1,z2,M,N)
% E is the tableau in the condensed form printed from simplextableau.m
% M is the number of rows
% N is the number of columns
% z1, z2 are the indices of columns and row respectively.

pivot=E(z2,z1);% pivot element

for i=1:M % do the pivoting including z_j-c_j row
    if((i~=z2)&(i~=1))
    % leave the 1st row and the leaving vector row
        for j=1:(N-1)
        % do for all the column except the entering and last
            if(j~=z1)
                val1 = E(z2,j)/pivot;
                val2 = E(i,z1)*val1;
                E(i,j)=E(i,j)-val2;
            end
        end
    end
end

E(z2,z1)=1/E(z2,z1);% making identity entry at the pivot entry

for i=1:(N-1)% dividing the leaving row with the pivot element
    if(i~=z1)
        E(z2,i)=E(z2,i)/pivot;
    end
end

for i=1:M % calculations pertaining to leaving column
    if((i~=z2)&(i~=1))
        E(i,z1)=E(i,z1)/(-1*pivot);
    end
end

return;
```

```
%Simplex algorithm when artificial variables are not required
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [opt_val,y,status]=Simplexmethod(cost,A,b)
% max/min c'*x
% subject to:
%          A*x <= b (b >=0)

%          x =>0

% input cost, A and b as column vector
% output: value, decision vector and status
% status=1 for feasible / alternate
% status=2 for unbounded


% function called by this function:simplextableau,
% findDepartindex,pivoting,remove


disp('Enter the LPP having less than equal constraints:')
MaxMin =input('Enter whether problem is Max/Min (0/1) :  ');
Num1 = input('Enter the Number of Constraints :  ');
Num2 = input('Enter the Number of Variables :  ');

% if the inputs are not passed when function is called
% then enter them here
if (nargin<3)
   [cost] = input('Enter Cost Matrix :  ');
   % This is a column vector of cost vector
   [b] =input('Enter b :  ');
   % This is a column vector of rhs vector
   [CoeffMatrix] = input('Enter the Coefficient Matrix :  ');
   A = CoeffMatrix;% constraint matrix, input row wise
end


if(MaxMin==1)
   % if the problem is min type then convert it to max
   cost = -1 * cost;
end
```

```
V = [1:Num2];% vector containing the index of nbv variables

% Var is the identity vector corresponding to slack variable
Var=eye(Num1);
A = [A Var]; % augment it with identity matrix

num_variables=size(A,2);% number of variables
B = eye(Num1);%basis matrix

[m,n]=size(A);
%U is vector of Basic Variables

k=1;
for i=1:Num1
    for j=1:n
        val=isequal(A(:,j),B(:,i));
        %  comparing the columns of A with identity matrix
        %and identifying the basis vectors
        if(val==1)
            U(1,k)=j;
            k=k+1;
        end
    end
end

num_variables=size(A,2);% number of variables
B = eye(Num1);%
C = zeros(num_variables-Num2,1);
cost=[cost;C];% total cost vector
Xb = inv(B) * b;% values of basic variables

[mbu,nbu]=size(U);
for i=1:nbu % Cb is cost vector corresponding to basic variables
    % U has the indices corresponding to basic variables
    Cb(i,1)=cost(U(1,i),1);
end

fval = Cb'*Xb;% Objective function value

Y = [];% calculation of Y_j=B^{-1}a_j
```

```
for j=1:n
    value = inv(B) * A(:,j);
    Y = [Y value];
end

% compute (Zj-Cj)
Z =zeros(1,n);
for j=1:n
    Z(1,j)=Cb'*Y(:,j)-cost(j);
end

[mbv,nbv]=size(V);
% this function constructs and prints the simplex tableau
% in the condensed form, where the first row corresponds
% to the indices of nonbasic variables and last
% column corresponds to indices of basic variables, last row
% corresponds to values of Z_j-c_j, first column corresponds
% to the values of basic variables
[E]=simplextableau(Y,Z,Xb,U,V,fval);

[M,N]=size(E);

% display tableau
disp('      Initial simplex tableau')
disp('     ----------------------------------')
disp(E)
unbounded=0;
status=0;
%repeat till optimality condition is satisfied or unbounded
% condition is met
while(status==0)
   [c]=find(E(M,2:(N-1)) >= 0);%checking the optimality criterion
   if (size(c,2)==nbv)
       status=1;
   else % go for one more iteration
       % check for unbounded solution
       for k=2:(N-1)
           if(E(M,k)<0)
               [r]=find(E(2:(M-1),k) <=0 );
               % checking the sign of y_{rj}
               if(size(r,1)==(M-2))
```

```
                % if length of r = m-2 i.e number of basic variables -1
                    disp('Problem has unbounded solution');
                    unbounded=1;
                    status=2;
                end
            end
        end
        %if the problem is not unbounded then proceed
        if unbounded~=1
            [minimum,index] = min(E(M,2:(N-1)));
            z1 = index(1)+1;
            % find the index having minimum (Zj-Cj)
            [z2] = findDepartindex(E,z1,M,N);
            % find the departing variable
            [E] = pivoting(E,z1,z2,M,N);
            % update tableau using pivoting
            % Swapping the indices of basic and non-basic variables
            x=U(:,z2-1);
            U(:,z2-1)=V(:,z1-1);
            V(:,z1-1)=x;
            % updating basic(non basic) variable index
            E(1,2:(N-1))=V;
            E(2:(M-1),N)=U';
            % display tableau
            disp('      Next simplex tableau ')
            disp('     ------------------------------')
            disp(E)
        end
    end
end
%if unbounded then  return
if(unbounded==1)
    y=[];
    opt_val=1e10;
    return;
end

opt_val=0;
if(unbounded==0)% optimal solution exist
    if(MaxMin==1)
        fprintf('Optimal Value :  %6.2f \n',-1 * E(M,1))
```

```
    opt_val=-1 * E(M,1);
else
    fprintf('Optimal Value :  %6.2f \n',E(M,1))
    opt_val=E(M,1);
end
status=1;

fprintf('Optimal Solution :  \n');
for i=1:Num2
    [x]=find(E(2:(M-1),N)==i);
    % tracing out the decision variables
    if(size(x,1)>0)
        y(i)=E(x(1)+1,1);
        fprintf('Value of variable %d  =  %6.2f \n',i,E(x(1)+1,1))
        %if it present in the optimal tableau then take the value
    else
        y(i)=0;
        fprintf('Value of variable %d  =  0 \n',i)
        %if decision variables are present at nonbasic variables
        %then its value is zero
    end
end
[c]=find(E(M,2:(N-1)) == 0);
if (size(c,2)>0)
    disp('Problem has Alternate Solution')
    [minimum,index] = min(E(M,2:(N-1)));
    z1 = index(1)+1;
    % find the index having minimum (Zj-Cj)
    [z2] = findDepartindex(E,z1,M,N);
    % find the departing variable
    [E] = pivoting(E,z1,z2,M,N);
    % update tableau using pivoting
    % Swapping basic and non-basic variables
    x=U(:,z2-1);
    U(:,z2-1)=V(:,z1-1);
    V(:,z1-1)=x;
    E(1,2:(N-1))=V;
    E(2:(M-1),N)=U';
    % display tableau
    disp('     Next simplex tableau ')
    disp('     ----------------------------')
```

```
        disp(E)
        disp('====================');
        fprintf('Alternate Optimal Solution :  \n');
        for i=1:Num2
        % tracing out the decision variables
            [x]=find(E(2:(M-1),N)==i);
            if(size(x,1)>0)
                y(i)=E(x(1)+1,1);
                fprintf('Value of variable %d  =  %6.2f \n',i,E(x(1)+1,1))
                %if it present in the optimal tableau then take the value
            else
                %fprintf('Value of variable %d  =  0 \n',i)
                y(i)=0;
                fprintf('Value of variable %d  =  0 \n',i)
                %if decision variables are present at nonbasic variables
                %then its value is zero
            end
        end
    end
end
```

```
%Simplex algorithm when artificial variables are required
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%code for two phase method
function[opt_val,y,status]=SimplexTwoPhasemethod
(cost,A,b,type_var,type,type_obj)
% min/max       c'*x
%subject to:    A*x (<=, =>, =) b
%               x (=>, <=, unrestricted) 0
% input cost, A and b as column vector
% output: value, decision vector and status
% status=2 for infeasible and unbounded
% status=1 for feasible
% function called by this function:simplextableau,
% findDepartindex,pivoting,remove
[Num1,Num2]=size(A);
disp('Enter the LPP having mixed constraints:')

if (nargin<6)
   % This is a column vector of cost vector
   type_obj =input('Enter whether problem is Max/Min (0/1) :  ');
end
if (nargin<5)
   for i=1:Num1
       fprintf('Constraint No. %d ',i)
       type(i) = input('Enter the  type <= or >= or = (1/2/3): ');
   end
end
if (nargin<4)
   for i=1:Num2
       fprintf('Variable No. %d \n',i)
       type_var(i) =input('Enter the type <= or >= or
       unrestricted(1/2/3):');
   end
end
if (nargin<3)
   [cost] = input('Enter Cost Matrix :  ');
   % This is a column vector of rhs vector
   [b] =input('Enter b :  ');
   [CoeffMatrix] = input('Enter the Coefficient Matrix :  ');
   A = CoeffMatrix;% constraint matrix, input row wise
end

MaxMin=type_obj;
```

```
index = find(b < 0);
if size(index,1)~=0
    b(index) = -b(index);
    A(index,:) = -A(index,:);
end
% if the variable is unrestricted then introduce one more variable
j=1;
for i=1:Num1
    if (type_var(i)==1)
        % if the variable is <=0 then multiply by -1,
        %change in the elements of A in c
        A(:,i)=-A(:,i);
        cost(i)=-cost(i);
    elseif (type_var(i)==3)
        A(:,Num2+j)=-A(:,i);
        %introduce variable with negative coefficient matrix
        %and cost vector
        % it will increase number  of variables by one
        cost(Num2+j)=-cost(j);
        j=j+1;
    end
end
if(MaxMin==1)
    cost = -1 * cost;
end
if isempty(type_var~=3)
    V = [1:Num2];
    % vector containing the indices of nbv variables
    k=Num2+1;
else
    V = [1:Num2+j-1];
    % vector containing the indices of nbv variables
    k=Num2+j;
end
% get the type of constraint
NumArtificalVar=zeros(1,Num1);
% number of artificial variables are atmost the number of constraints

% Var is the identity vector corresponding to slack or
% artificial variable
for i=1:Num1
```

```
    Var=zeros(Num1,1);
    if(type(i)==1)
        Var(i,1)=1;  % constraint of type <=
        A = [A Var]; % augment it with identity matrix
    else
        if(type(i)==2)
            Var(i,1)=-1;  % constraint of type >=
            NumArtificalVar(1,i)=1; % introduce artificial variable
            % surplus variable corresponding to ith constraint
            V(1,k)=Num2+i;
            k=k+1;
            A = [A Var];%augment it with identity matrix
        else
            Var(i,1)=1;  % constraint of type =
            NumArtificalVar(1,i)=1;%introduce artificial variable
        end
    end
end
% add artificial variables
k=1;
[m,n]=size(A);
Art=[];% empty matrix for artificial variable
for i=1:Num1
    if(NumArtificalVar(1,i)==1)
        Var=zeros(Num1,1);
        Var(i,1)=1;
        A = [A Var];
        Art(1,k)=n+k;
        k=k+1;
    end
end
num_variables=size(A,2);% number of variables
B = eye(Num1);%
C = zeros(num_variables-Num2,1);
cost=[cost;C];% total cost vector

[m,n]=size(A);
% Basic Variables U
k=1;
if Num1==1 & type==3
    U(1,k)=n;
```

```
    else
        for i=1:Num1
            for j=1:n
                val=isequal(A(:,j),B(:,i));
                %  comparing the columns of A with identity matrix
                %and identifying the basis vectors
                if(val==1)
                    U(1,k)=j;
                    k=k+1;
                end
            end
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Phase-I if the constraints are of type ==2,3
    % and normal simplex if type==1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    cost1=zeros(n,1);

    [mArt,nArt]=size(Art);

    if nArt~=0
        for i=1:nArt
        % cost assigned to artificial variable
            cost1(Art(1,i),1)=-1;
        end
    else
        cost1=cost;
    end
    Xb = inv(B) * b;% value of basic variables
    [mbu,nbu]=size(U);
    for i=1:nbu
        if nArt~=0
            Cb(i,1)=cost1(U(1,i),1);
        else
            Cb(i,1)=cost(U(1,i),1);
        end
    end
    fval = Cb'*Xb;% Objective function value
    Y = [];% calculation of Y_j=B^{-1}a_j
    for j=1:n
```

```
    value = inv(B) * A(:,j);
    Y = [Y value];
end
% compute (Zj-Cj)
Z =zeros(1,n);
for j=1:n
    Z(1,j)=Cb'*Y(:,j)-cost1(j);
end
[mbv,nbv]=size(V);
% this function constructs and prints the simplex tableau
% in the condensed form, where the first row corresponds
% to the indices of nonbasic variables and last
% column corresponds to indices of basic variables, last row
% corresponds to values of Z_j-c_j, first column corresponds
% to the values of basic variables
[E]=simplextableau(Y,Z,Xb,U,V,fval);
[M,N]=size(E);
% display tableau
disp('      Initial simplex tableau - Phase I')
disp('     ---------------------------------')
disp(E)
infeasible=0;
while(1)
    [c]=find(E(M,2:(N-1)) >= 0);%checking the optimality criterion
    if (size(c,2)==nbv)  % End Phase I
        fval = E(M,1);
        if(abs(fval)>=0.00001)% stop with this epsilon tolerance
        % when artificial variable comes in picture
            if ~isempty(find(type~=1))
                infeasible=1;
                disp('LPP is infeasible')
                y=[];
                opt_val=-1000000;
                status=2;
                break;
            else
            %if the problem deals with slack variables
            % only then print the optimal  solution
                if(MaxMin==1)
                    fprintf('Optimal Value:%6.2f \n',-1 * E(M,1))
                    opt_val=-1 * E(M,1);
```

```
                else
                    fprintf('Optimal Value :  %6.2f \n',E(M,1))
                    opt_val=E(M,1);
                end
                fprintf('Optimal Solution :  \n');
                for i=1:Num2
                    % tracing out the decision variables
                    [x]=find(E(2:(M-1),N)==i);
                    if(size(x,1)>0)
                    % value of decision variable
                        y(i)=E(x(1)+1,1);
                        %if it present in the optimal tableau then
                        %take the value from the optimal tableau
            fprintf('Value of variable %d=%6.2f \n',i,E(x(1)+1,1))
                    else
                        %fprintf('Value of variable %d  =  0 \n',i)
                        y(i)=0;
                        %if decision variables are present at
                        %nonbasic variables then its value is zero
                        fprintf('Value of variable %d  =  0 \n',i)
                    end
                end
            end
        end
        break;
    else % go for one more iteration
        [minimum,index] = min(E(M,2:(N-1)));
        % find the index having minimum (Zj-Cj)
        z1 = index(1)+1;
        % find the departing variable
        [z2] = findDepartindex(E,z1,M,N);
        % update tableau using pivoting
        [E] = pivoting(E,z1,z2,M,N);
        % Swapping the indices basic and non-basic variables
        x=U(:,z2-1);
        U(:,z2-1)=V(:,z1-1);
        V(:,z1-1)=x;
        E(1,2:(N-1))=V;
        E(2:(M-1),N)=U';
        % display tableau
        disp('      Next simplex tableau - Phase I')
```

```
        disp('      ------------------------------------------')
        disp(E)
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   if the problem is of type=2 or 3 then go for  Phase-II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(infeasible==0)% if not infeasible then move to phase II
    Xb = E(2:(M-1),1);% value of bv's
    % remove the artificial variable column from the tableau
    %Here it is assumed that all artificial variables are zero
    %as nonbasic variables. If some artificial variable is zero
    %as a basic variable then ‘‘exchange" is to be done before
    %going to Phase II. This part can be coded separately.
    for i=1:nArt
        [a]=find(E(1,:)==Art(1,i));
        if(size(a,2)~=0)
            [E]=remove(E,a(1));  % remove from E
            [b]=find(V(1,:)==Art(1,i));
            [V]=remove(V,b(1));  % remove from V
        end
    end
    [M,N]=size(E);
    [mbu,nbu]=size(U);
    Cb=zeros(nbu,1);
    for i=1:nbu
        Cb(i,1)=cost(U(1,i),1);
    end
    % compute new (Zj-Cj)
    for j=2:(N-1)
        Y = E(2:(M-1),j);
        E(M,j) = Cb'*Y-cost(E(1,j),1);
    end
    % compute initial value of phase II
    fval = Cb'*Xb;
    E(M,1)=fval; % update E matrix
    [mbv,nbv]=size(V);
    % display tableau
    disp('      Initial simplex tableau - Phase II')
    disp('    --------------------------------------')
    disp(E)
```

```
unbounded=0;
while(1)
    % do iteration until all z_j-c_j are not >= 0
    [c]=find(E(M,2:(N-1)) >= 0);
    if (size(c,2)==nbv)  % optimal solution
        % check for alternate solution
        [c]=find(E(M,2:(N-1)) == 0);
        if (size(c,2)>0)
            disp('Problem has Alternate Solution')
        end
        break;
    else
        [minimum,index] = min(E(M,2:(N-1)));
        % find the index having minimum (Zj-Cj)
        z1 = index(1)+1;
        % find the departing variable
        [z2] = findDepartindex(E,z1,M,N);
        % update tableau using pivoting
        [E] = pivoting(E,z1,z2,M,N);
        % Swapping the indices of basic and non-basic variables
        x=U(:,z2-1);
        U(:,z2-1)=V(:,z1-1);
        V(:,z1-1)=x;
        E(1,2:(N-1))=V;
        E(2:(M-1),N)=U';
        % display tableau
        disp('      Next simplex tableau - Phase II')
        disp('     ---------------------------------')
        disp(E)
        % check for unbounded solution
        for k=2:(N-1)
            if(E(M,k)<0)
                % if length of r=m-2 i.e no of Basic variables -1
                % checking the sign of y_{rj}
                [r]=find(E(2:(M-1),k) <=0 );
                if(size(r,1)==(M-2))
                    disp('Problem has unbounded solution');
                    unbounded=1;
                    status=2;
                    break;
                end
```

```
            end
        end
        if(unbounded==1)
            y=[];
            opt_val=1e10;
            break;
        end
    end
end
opt_val=0;
% unbounded is zero implies that optimal solution is achieved
if(unbounded==0)
    if(MaxMin==1)
        fprintf('Optimal Value :  %6.2f \n',-1 * E(M,1))
        opt_val=-1 * E(M,1);
    else
        fprintf('Optimal Value :  %6.2f \n',E(M,1))
        opt_val=E(M,1);
    end
    status=1;
    fprintf('Optimal Solution :  \n');
    for i=1:Num2
        % tracing out the decision variables
        [x]=find(E(2:(M-1),N)==i);
       if(size(x,1)>0)
        y(i)=E(x(1)+1,1);
        %if it present in the optimal tableau then take the value
        fprintf('Value of variable %d  =  %6.2f \n',i,E(x(1)+1,1))
       else
        %fprintf('Value of variable %d  =  0 \n',i)
        y(i)=0;
        %if decision variables are present at nonbasic variables
        %then its value is zero
        fprintf('Value of variable %d  =  0 \n',i)
       end
      end
    end
end
```

19.3 Dual Simplex Algorithm

```
%The dual simplex method can be used to solve
%any LPP provided we have an initial basic solution
%for which all (z_j-c_j)>=0, in particular we may take
%the LPP in the form
% max    c'*x
%subject to:
%        A*x (<=, =>) b
%        x =>0
%with c <=0

disp('Enter the LPP to be solved by dual simplex :')

Num1 =input('Enter the Number of Constraints :  ');

Num2 = input('Enter the Number of Variables :  ');

[CoeffMatrix] = input('Enter the Coefficient Matrix : ');
% enter the coefficient of each of the variable even
% though it may be zero
[b] = input('Enter b :  ');  % This is a column vector
[cost] = input('Enter Cost Matrix :  '); % This is a column vector

for i=1:Num1
    fprintf('Constraint No. %d ',i)
    type = input('Enter the  type <= or >=   (1/2) :  ');
    if(type==1)
        CoeffMatrix(i,:)=-1* CoeffMatrix(i,:);
        b(i)=-1*b(i);
    end
end
Surplus = -1*eye(Num1);
A = [CoeffMatrix Surplus];
B =-1*eye(Num1);
for i=1:Num1
    U(1,i) = Num2+i;
end
V = [1:Num2];
C = zeros(Num1,1);
cost=[cost;C];
```

```
[m,n]=size(A);
Xb = inv(B) * b;
fval = C'*Xb;
Y = [];
for j=1:n
    value = inv(B) * A(:,j);
    Y = [Y value];
end
% compute (Zj-Cj) which are bound to be >=0
% for the stated form of the problem
Z =zeros(1,n);
for j=1:n
    Z(1,j)=C'*Y(:,j)-cost(j);
end
[mbv,nbv]=size(U);
% constuct tableau in condensed form
[E]=Dualsimplextableau(Y,Z,Xb,U,V,fval);
[M,N]=size(E);
% display tableau
disp('      Initial dual simplex tableau')
disp('-----------------------')
disp(E)
infeasible=0;
while(1)
    [c]=find(E(2:(M-1),1)>=0);
    if (size(c,1)~=nbv)
    % solution is not optimal solution
    % go for one more iteration
        [minimum,index] = min(E(2:(M-1),1));
        % find the index having minimum Xb
        z2 = index(1)+1;
        % find the departing variable
        [z1] = Dualdepartindex(E,z2,M,N);
        % update tableau using pivoting
        [E] = pivoting(E,z1,z2,M,N);
        % Swapping the indices basic and non-basic variables
        x=U(:,z2-1);
        U(:,z2-1)=V(:,z1-1);
        V(:,z1-1)=x;
        E(1,2:(N-1))=V;
        E(2:(M-1),N)=U';
```

```
        % display tableau
        disp('      Next simplex tableau')
        disp('     -------------------------')
        disp(E)
       % check for the infeasibility of the problem
         for k=2:(M-1)
           if(E(k,1)<0)
                [r]=find(E(k,2:(N-1)) >=0 );
                  if(size(r,2)==(N-2))
                      disp('Problem is infeasible');
                          infeasible=1;
                      break;
                  end
            end
         end
        if(infeasible==1)
            break;
        end
    else
        break;
    end
end
if(infeasible==0) % print the optimal solution
    fprintf('Optimal Value :  %6.2f \n',E(M,1))
    fprintf('Optimal Solution :  \n');
    for i=1:Num2
        [x]=find(E(2:(M-1),N)==i);
        if(size(x,1)>0)
            fprintf('Value of variable %d = %6.2f\n',i,E(x(1)+1,1))
        else
            fprintf('Value of variable %d  =  0 \n',i)
        end
    end
end
```

```
% This function displays the dual simplex tableau in
% the condensed form and prints it, where the first row
% corresponds to the indices of nonbasic variables and the
% last column corresponds to the indices of basic variables.
% Last row corresponds to the values of Z_j-c_j, and the first
% column corresponds to the values of the basic variables.

function [E] = Dualsimplextableau(Y,Z,Xb,U,V,fval)

% Y is the set of columns of the simplex tableau which correspond
% to non basic variables
% fval is the value of the objective function corresponding
% to the current basic solution Xb.
% U(V) is a vector of indices corresponding to basic(non basic)variables
% Z is a vector whose entries correspond to the values of zj-cj


[m,n]=size(Y);
[M,N]=size(V);
E=zeros(m+2,N+2);

for i=1:N
    E(2:(m+1),i+1)=Y(:,V(i));
    E(m+2,i+1)=Z(V(i));
end

E(1,2:(N+1))=V; E(2:(m+1),N+2)=U'; E(2:(m+1),1)=Xb; E(m+2,1)=fval;
return;
```

```
function [z1] = Dualdepartindex(E,z2,M,N)
% This function computes the departing variable index
%z2 is the index of the entering row

k=1; for i=2:(N-1)
    if(E(z2,i)<0)
        ratio(k) = E(M,i)/E(z2,i);
    else
        ratio(k)=-inf;
    end
    k=k+1;
end

[r]=find(ratio==max(ratio)); z1=r(1)+1; return;
```

19.4 Assignment Problem: Hungarian Method

```
%**********TO SOLVE BALANCED ASSIGNMENT PROBLEM**********
c=input('\nEnter cost matrix');% enter row wise
[m,n]=size(c);
%m-no of jobs
%n-no of persons, here m and n are equal
c1=c;
% To subtract row minima from each row
for i=1:n
    min1=min(c1(i,:));
    c1(i,:)=c1(i,:)-min1;
end
% To subtract column minima from each  column
for j=1:n
    min1=min(c1(:,j));
    c1(:,j)=c1(:,j)-min1;
end disp(c1);
fprintf(1,'1(0) is displayed if corresponding row or
column is ticked(unticked)\n');

% opt is a variable which represents number of independent zeros
opt=0;
%loop executes until number of zeros=n
%-10(-20) denotes encircled(crossed) zero
while(opt<n)
    flag_m=1;
    %stop when either all zeros are encircled or crossed
    while(flag_m)
        flag_m=0;
        flag=1;
        % stop when there is no row left with a single zero
        while(flag)
            flag=0;
            for i=1:n
                if numel(find(c1(i,:)==0))==1
                    j=find(c1(i,:)==0);
                    c1(i,j)=-10;
                    c1(find(c1(:,j)==0),j)=-20;
                end
            end
```

```
            for i=1:n
                if numel(find(c1(i,:)==0))==1
                    flag=1;
                end
            end
        end
        flag=1;
        % stop when there is no column left with a single zero
        while(flag)
            flag=0;
            for j=1:n
                if numel(find(c1(:,j)==0))==1
                    i=find(c1(:,j)==0);
                    c1(i,j)=-10;
                    c1(i,find(c1(i,:)==0))=-20;
                end
            end
            for j=1:n
                if numel(find(c1(:,j)==0))==1
                    flag=1;
                end
            end
        end
        for i=1:n %search for a row with a single zero
            if numel(find(c1(i,:)==0))==1
                flag_m=1;
            end
        end
        % to encircle and cross zeros if there is no row or
        % column left with a single zero but there are zeros left
        if numel(find(c1==0))~=0 & flag_m==0
            [i,j]=find(c1==0);
            i=i(1);
            j=j(1);
            c1(i,j)=-10;
            c1(find(c1(:,j)==0),j)=-20;
            c1(i,find(c1(i,:)==0))=-20;
            flag_m=1;
        end
    end
    % u(v) takes value 1 if row is ticked & 0 otherwise
```

```
u=zeros(1,n);
% to tick rows which do not have encircled zero
for i=1:n
    if numel(find(c1(i,:)==-10))==0
        u(i)=1;
    end
end
v=zeros(1,n);
flag=1;
while(flag) % stop when chain of ticking is completed
    flag=0;
    % to tick columns having zeros in ticked rows
    for j=1:n
        if numel(find((c1(:,j)==-20) & u'==1))~=0
            v(j)=1;
        end
        if numel(find((c1(:,j)==-10) & u'==1))~=0
            v(j)=1;
        end
    end
    % to tick rows that have encircled zeros in ticked columns
    for i=1:n
        if numel(find(c1(i,:)==-10 & v==1))~=0
            u(i)=1;
        end
    end
    for j=1:n
        if numel(find((c1(:,j)==-10) & u'==1))~=0 & v(j)==0
            flag=1;
        end
        if numel(find((c1(:,j)==-20) & u'==1))~=0 & v(j)==0
            flag=1;
        end
    end
end
fprintf(1,'\n\n');
c2=c1;
c1(find(c1==-10))=0;
c1(find(c1==-20))=0;
% to calculate min no of lines required to cover all zeros
opt=numel(find(u==0))+ numel(find(v==1));
```

```
        disp([c1 u']);
        disp(v);
        if opt<n % if no of independent zeros < n
            % to calculate min of uncovered elements
            min1=max(abs(c1));
            min1=min1(1);
            for i=1:n
                for j=1:n
                    if u(i)==1 & v(j)==0
                        if min1 > c1(i,j)
                            min1=c1(i,j);
                        end
                    end
                end
            end
            for i=1:n
                for j=1:n
                    % to add min at the intersection of two lines
                    if u(i)==0 & v(j)==1
                        c1(i,j)=c1(i,j)+min1;
                    end
                    % to subtract min from uncovered elements
                    if u(i)==1 & v(j)==0
                        c1(i,j)=c1(i,j)-min1;
                    end
                end
            end
        end
    end
end
% to calculate optimal value and print result
opt_val=0;
fprintf(1,'\n Optimal job assignment is:')
for i=1:m
    j=find(c2(i,:)==-10);
    if j<=n
        opt_val=opt_val+c(i,j);
        fprintf(1,'\n Job %d -> Person%d',i,j);
    end
end
fprintf(1,'\n Optimal value is %d\n\n',opt_val);
```

19.5 Branch and Bound Method

```
% code for branch and bound method (BnB)
% BnB  finds the solution for integer programming problem.
function [x,val,status]=bnb1()
%call from command window: [x,val,status]=bnb1()
A=[-1 3;
    7 1;
    1 0;
    0 1];
b=[6;
    35;
    7;
    7];
cost=[-7;
    -9];%take the negative for maximization problems

e=2^-24;
type_var=2;
type=1;
[m,n]=size(A);
M=[type_var*ones(m,1)];%typr_var is 2 for >=0 variables
%(<= or >= or unrestricted  (1/2/3)).
%type = vector of 1 or 2 or 3,i.e, <= or >= or =  (1/2/3)
N=[type*ones(m,1)];
P=[1,2];
bound=inf; % the initial bound is set to +ve infinity
[x0,val0,status0]=Simplexbnb(cost,A,b,M,N);
% a recursive function that processes the BB tree
[x,val,status,bound]=branch(cost,A,b,x0,val0,M,N,e,bound,P);
val=-val;


function [xx,val,status,bb]=branch(cost,A,b,x,v,M,N,e,bound,P)
% x is an initial solution and v is the
% corresponding objective function value
% solve the corresponding LP model with the integarily
% constraints removed
[x0,val0,status0]=Simplexbnb(cost,A,b,M,N);

% if the solution is not feasible or the value of the
```

```
  % objective function is higher than the current bound
  % return with the input initial solution
  if status0<=0 | val0 > bound
      xx=x; val=v; status=status0; bb=bound;
      return;
  end
  % if the integer-constraint variables turned to be
  % integers within the input tolerance return
  ind=find( abs(x0(P)-round(x0(P)))>e ); if isempty(ind)
      status=1;
      % this solution is better than the current solution
      % hence replace
      if val0 < bound
          x0(P)=round(x0(P));
          xx=x0;
          val=val0;
          bb=val0;
      else
          xx=x;  % return the input solution
          val=v;
          bb=bound;
      end
      return
end
% if we come here this means that the solution of
% the LP relaxation is feasible and gives a less value
% than the current bound but some of the integer-constraint
% variables are not integers. Therefore we pick the
% first one that is not integer and form two LP problems
% and solve them recursively by calling the same function(branching)
% first LP problem with the added constraint that
% Xi <= floor(Xi) , i=ind(1)
br_var=P(ind(1));
br_value=x(br_var);
if isempty(A)
    [r c]=size(Aeq);
else
    [r c]=size(A);
end
A1=[A ; zeros(1,c)];
A1(end,br_var)=1;
```

```
b1=[b;floor(br_value)];
%M=[M];
N1=[N;1];

% second LP problem with the added constraint that
% Xi >= ceil(Xi) , i=ind(1)
A2=[A ;zeros(1,c)];
A2(end,br_var)=-1;
b2=[b; -ceil(br_value)];
%M=[M;2];
N2=[N;2];

% solve the first LP problem
[x1,val1,status1,bound1]=branch(cost,A1,b1,x0,val0,M,N1,e,bound,P);
status=status1;
% if the solution was successfull and gives a better bound
if status1 >0 & bound1<bound
   xx=x1;
   val=val1;
   bound=bound1;
   bb=bound1;
else
    xx=x0;
    val=val0;
    bb=bound;
end
% solve the second LP problem
[x2,val2,status2,bound2]=branch(cost,A2,b2,x0,val0,M,N2,e,bound,P);
% if the solution was successfull and gives a better bound
if status2 >0 & bound2<bound
    status=status2;
    xx=x2;
    val=val2;
    bb=bound2;
end
```

```
% code for branch and bound. remaining functions are
% similar to two phase method
function [y,opt_val,status]=Simplexbnb(cost,A,b,type_var,type)
%type_var=variable type
%type=type of constraints
% min c'*x    subject to:   A*x (<=, =>, =) b
%            x =>0
%status=-2 for infeasible and unbounded
% status=1 for feasible
MaxMin =1;

cost=-cost;

[Num1,Num2]=size(A);

index = find(b < 0);
if size(index,1)~=0
    b(index) = -b(index);
    A(index,:) = -A(index,:);
end

j=1;%
for i=1:Num2
    % if the variable is <=0 then multiply
    % by -1, the coeff in A in c
    if (type_var(i)==1)
        A(:,i)=-A(:,i);
        cost(i)=-cost(i);
    elseif (type_var(i)==3)
        %introduce variable with negative coefficient
        %matrix and cost vector
        A(:,Num2+j)=-A(:,i);
        % it will increase no  of variables by one
        cost(Num2+j)=-cost(j);
        j=j+1;
    end
end

if isempty(type_var~=3)
    V = [1:Num2];
    % vector containing the indices of nbv variables
```

```
    k=Num2+1;
else
    V = [1:Num2+j-1];
    % vector containing the indices of nbv variables
    k=Num2+j;
end
% get the type of constraint
NumArtificalVar=zeros(1,Num1);
% number of artificial variables are atmost the number of constraints

% Var is the identity vector corresponding to slack or
% artificial variable
for i=1:Num1
    Var=zeros(Num1,1);
    if(type(i)==1)
        Var(i,1)=1;  % constraint of type <=
        A = [A Var]; % augment it with identity matrix
    else
        if(type(i)==2)
            Var(i,1)=-1;  % constraint of type >=
            NumArtificalVar(1,i)=1; % introduce artificial variable
            % surplus variable corresponding to ith constraint
            V(1,k)=Num2+i;
            k=k+1;
            A = [A Var];%augment it with identity matrix
        else
            Var(i,1)=1;  % constraint of type =
            NumArtificalVar(1,i)=1;%introduce artificial variable
        end
    end
end
% add artificial variables
k=1; [m,n]=size(A);
Art=[];% empty matrix for artificial variable
for i=1:Num1
    if(NumArtificalVar(1,i)==1)
        Var=zeros(Num1,1);
        Var(i,1)=1;
        A = [A Var];
        Art(1,k)=n+k;
        k=k+1;
```

```
    end
end
num_variables=size(A,2);% number of variables
B = eye(Num1);%
C = zeros(num_variables-Num2,1);
cost=[cost;C];% total cost vector

[m,n]=size(A);
% Basic Variables U
k=1; for i=1:Num1
    for j=1:n
        val=isequal(A(:,j),B(:,i));
        %  comparing the columns of A with identity matrix
        %and identifying the basis vectors
        if(val==1)
            U(1,k)=j;
            k=k+1;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Phase-I if the constraints are of type ==2,3
% and normal simplex if type==1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cost1=zeros(n,1);
[mArt,nArt]=size(Art);

if nArt~=0
    for i=1:nArt
    % cost assigned to artificial variable
        cost1(Art(1,i),1)=-1;
    end
else
    cost1=cost;
end
Xb = inv(B) * b;% value of basic variables

[mbu,nbu]=size(U);
for i=1:nbu
    if nArt~=0
        Cb(i,1)=cost1(U(1,i),1);
```

```
    else
        Cb(i,1)=cost(U(1,i),1);
    end
end
fval = Cb'*Xb;% Objective function value
Y = [];% calculation of Y_j=B^{-1}a_j
for j=1:n
    value = inv(B) * A(:,j);
    Y = [Y value];
end
% compute (Zj-Cj)
Z =zeros(1,n);

for j=1:n
    Z(1,j)=Cb'*Y(:,j)-cost1(j);
end
[mbv,nbv]=size(V);
% this function constructs and prints the simplex tableau
% in the condensed form, where the first row corresponds
% to the indices of nonbasic variables and last
% column corresponds to indices of basic variables, last row
% corresponds to values of Z_j-c_j, first column corresponds
% to the values of basic variables
[E]=simplextableau(Y,Z,Xb,U,V,fval);

[M,N]=size(E);
% display tableau
disp('      Initial simplex tableau - Phase I')
disp('-----------------------------------')
disp(E)
infeasible=0;
while(1)
    [c]=find(E(M,2:(N-1)) >= 0);%checking the optimality criterion
    if (size(c,2)==nbv)  % End Phase I
        fval = E(M,1);
        if(abs(fval)>=0.00001)% stop with this epsilon tolerance
        % with artificial variable comes in picture
            if ~isempty(find(type~=1))
                infeasible=1;
                disp('LPP is infeasible')
                y=[];
```

```
                opt_val=-1000000;
                status=-2;
                break;
            else
            %if the problem deals with slack variables
            % only then print the optimal  solution
                if(MaxMin==1)
                    fprintf('Optimal Value:%6.2f \n',-1 * E(M,1))
                    opt_val=-1 * E(M,1);
                else
                    fprintf('Optimal Value :  %6.2f \n',E(M,1))
                    opt_val=E(M,1);
                end
                fprintf('Optimal Solution :  \n');
                for i=1:Num2
                    % tracing out the decision variables
                    [x]=find(E(2:(M-1),N)==i);
                    if(size(x,1)>0)
                    % value of decision variable
                        y(i)=E(x(1)+1,1);
                        %if it present in the optimal tableau then
                        %take the value from the optimal tableau
            fprintf('Value of variable %d=%6.2f \n',i,E(x(1)+1,1))
                    else
                        %fprintf('Value of variable %d  =  0 \n',i)
                        y(i)=0;
                        %if decision variables are present at
                        %nonbasic variables then its value is zero
                        fprintf('Value of variable %d  =  0 \n',i)
                    end
                end
            end
        end
        break;
    else % go for one more iteration
        [minimum,index] = min(E(M,2:(N-1)));
        % find the index having minimum (Zj-Cj)
        z1 = index(1)+1;
        % find the departing variable
        [z2] = findDepartindex(E,z1,M,N);
        % update tableau using pivoting
```

```
        [E] = pivoting(E,z1,z2,M,N);
        % Swapping the indices basic and non-basic variables
        x=U(:,z2-1);
        U(:,z2-1)=V(:,z1-1);
        V(:,z1-1)=x;
        E(1,2:(N-1))=V;
        E(2:(M-1),N)=U';
        % display tableau
        disp('      Next simplex tableau - Phase I')
        disp('     -------------------------------')
        disp(E)
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   if the problem is of type=2 or 3 then go for  Phase-II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(infeasible==0)% if not infeasible then move to phase II
   Xb = E(2:(M-1),1);% value of bv's
   % remove the artificial variable column from the tableau
   %Here it is assumed that all artificial variables are zero
   %as nonbasic variables. If some artificial variable is zero
   %as a basic variable then ``exchange" is to be done before
   %going to Phase II. This part can be coded separately.
   for i=1:nArt
       [a]=find(E(1,:)==Art(1,i));
       if(size(a,2)~=0)
           [E]=remove(E,a(1));  % remove from E
           [b]=find(V(1,:)==Art(1,i));
           [V]=remove(V,b(1));  % remove from V
       end
   end
   [M,N]=size(E);
   [mbu,nbu]=size(U);
   Cb=zeros(nbu,1);
   for i=1:nbu
       Cb(i,1)=cost(U(1,i),1);
   end
   % compute new (Zj-Cj)
   for j=2:(N-1)
       Y = E(2:(M-1),j);
       E(M,j) = Cb'*Y-cost(E(1,j),1);
```

```
end
% compute initial value of phase II
fval = Cb'*Xb;
E(M,1)=fval; % update E matrix
[mbv,nbv]=size(V);
% display tableau
disp('      Initial simplex tableau - Phase II')
disp('     ----------------------------------')
disp(E)
unbounded=0;
while(1)
    % do iteration until all z_j-c_j are not >= 0
    [c]=find(E(M,2:(N-1)) >= 0);
    if (size(c,2)==nbv)  % optimal solution
        % check for alternate solution
        [c]=find(E(M,2:(N-1)) == 0);
        if (size(c,2)>0)
            disp('Problem has Alternate Solution')
        end
        break;
    else
        [minimum,index] = min(E(M,2:(N-1)));
        % find the index having minimum (Zj-Cj)
        z1 = index(1)+1;
        % find the departing variable
        [z2] = findDepartindex(E,z1,M,N);
        % update tableau using pivoting
        [E] = pivoting(E,z1,z2,M,N);
        % Swapping the indices of basic and non-basic variables
        x=U(:,z2-1);
        U(:,z2-1)=V(:,z1-1);
        V(:,z1-1)=x;
        E(1,2:(N-1))=V;
        E(2:(M-1),N)=U';
        % display tableau
        disp('      Next simplex tableau - Phase II')
        disp('     ---------------------------------')
        disp(E)
        % check for unbounded solution
        for k=2:(N-1)
            if(E(M,k)<0)
```

```
                % if length of r=m-2 i.e no of Basic variables -1
                % checking the sign of y_{rj}
                [r]=find(E(2:(M-1),k) <=0 );
                if(size(r,1)==(M-2))
                    disp('Problem has unbounded solution');
                    unbounded=1;
                    status=2;
                    break;
                end
            end
        end
        if(unbounded==1)
            y=[];
            opt_val=1e10;
            break;
        end
    end
end
opt_val=0;
% unbounded is zero implies that optimal solution is achieved
if(unbounded==0)
    if(MaxMin==1)
        fprintf('Optimal Value :  %6.2f \n',-1 * E(M,1))
        opt_val=-1 * E(M,1);
    else
        fprintf('Optimal Value :  %6.2f \n',E(M,1))
        opt_val=E(M,1);
    end
    status=1;
    fprintf('Optimal Solution :  \n');
    for i=1:Num2
        % tracing out the decision variables
        [x]=find(E(2:(M-1),N)==i);
      if(size(x,1)>0)
        y(i)=E(x(1)+1,1);
        %if it present in the optimal tableau then take the value
        fprintf('Value of variable %d  =  %6.2f \n',i,E(x(1)+1,1))
      else
        %fprintf('Value of variable %d  =  0 \n',i)
        y(i)=0;
        %if decision variables are present at nonbasic variables
```

```
            %then its value is zero
            fprintf('Value of variable %d  =  0 \n',i)
        end
      end
    end
end
```

19.6 Golden Section Rule

```
function x = golden_section(xl, xu, eps,y)
% Golden Section - This function calculates the minimum value of
% a unimodal function of 1-variable using Golden Section Rule. Here,
% the objective function taken is a polynomial of degree n, but the
% code can be appropriately modified for general case as well.
% function is Min a_nx^n + a_{n-1}x^{n-1} .....+ a_0
% Inputs:
%   xl  : Lower bound of variable
%   xu  : Upper bound of variable
%   eps : Epsilon Parameter (For stopping the algorithm)
%   m=input('the order of poly =');
%   for i=1:m+1
%     y(i)=input('the coeff of poly starting from constt =');
%   end
% for eg. min x^2 in [-5,15] cal following
% x = golden_section(-5, 15, 0.01,[0;0;1])

xp = xu - 0.618 * (xu-xl);
xq = xl + 0.618 * (xu-xl);
k = 1;
Ep = mypoly(y,xp);% or any other function for eg, feval('sin',xp)
Eq = mypoly(y,xq);
I1 = xu - xl;
while (I1 > eps)
    if ( Ep <= Eq)
        xu = xq;
        xq = xp;
        xp = xu - 0.618*(xu - xl);
    else
        xl = xp;
        xp = xq;
        xq = xl + 0.618*(xu - xl);
    end
    k = k + 1;
    Ep = mypoly(y,xp);
    Eq = mypoly(y,xq);
    I1 = xu - xl;
end
```

```
x = (xu + xl)/2;


function x=mypoly(y,a)

for i=1:length(y)-1
    b(i)=y(i+1)*a^i;
end
x=sum(b)+y(1);
return;
```

19.7 Steepest Descent Method

```
% Min f(x), x in R^n.
% x_k+1 = x_k + alpha_k*d_k, d_k=-grad(f(x_k))/norm(grad(f(x_k)));
% h(alpha_k)=Min_{alpha_k >= 0}f(x_k+alpha+kd_k)

function x = steepest_descent(Q, b, c, eps)
%steepest_descent - This function is used to minimize
%a quadratic function of n-variables using Steepest Descent method
%The objective function is
%Min 0.5*x'Qx - b'x + c (with Q positive semi-definite)
% for x1^2+x1x2+x2^2+x1+x2: Q= [ 2 1;1 2];

% Inputs:
%   Q, b : Coefficients in the objective function
%   eps  : Epsilon parameter (For stopping rule)

% Output:
%   x    : Optimal Solution

% Q is the Hessian of f
%(here Q corresponds to Hessian of the quadratic function)

n = length(b);
x = zeros(n,1); % x_0=[0 0 ... 0]'
k = 0;

gradf = Q*x - b; % g_0
d = - gradf / norm(gradf); % d_0

while(norm(gradf)>eps)
%     Calculating h(alpha)=C(1)*alpha_k^2 + C(2)*alpha_k + C(3)
% C is vector of coefficients of polynomial
C = [0.5*(d'*Q*d) (0.5*d'*Q*x + 0.5*x'*Q*d - b'*d) ...
       (0.5*x'*Q*x - b'*x + c)];
% for general function golden section rule can be used to
% calculate the min, whereas for the quadratic case
% we can obtain it directly
   % alpha = golden_section(0,10000,0.1,C)
   alpha = - C(2)/(2*C(1));
```

```
%     x_k+1 = x_k + alpha_k*d_k
    x = x + alpha*d;
    k = k+1;

%     Calculating g_k+1
    gradf = Q*x - b;

%     d_k+1 = -g_k+1 / ||g_k+1||
    d = - gradf / norm(gradf);
end
```

19.8 Conjugate Gradient Method

```
function x = conjugate_gradient(Q, b)
% conjugate_gradient - This function is used to minimize a
% positive definite quadratic form of n-variables using
% Conjugate Gradient method
% The objective function is
%       Min 0.5*x'Qx - b'x, Q positive definite
%
% Inputs:
%   Q, b : Coefficients in the objective function
% Output:
%   x    : Optimal Solution
% Fletcher-Reeves method is used for the general convex function f

n = length(b);
x = zeros(n,1); % x_0=[0 0 ... 0]'

g = Q*x - b;      % g_0=grad(f)
d = -g;           % d_0 = -g_0
k = 0;
while ((k < n) && (norm(g) ~= 0))
%     alpha_k = -(g_k'*d_k) / (d_k'*Q*d_k)
    alpha = -(g'*d) / (d'*Q*d);

%     x_k+1 = x_k + alpha_k*d_k
    x = x + alpha*d;
    k = k + 1;
%     Calculating g_k+1
    g = Q*x - b;

    if (norm(g) == 0)
        break;
    end
%     beta_k = (g_k+1'*Q*d_k) / (d_k'*Q*d_k)
    beta = (g'*Q*d) / (d'*Q*d);

%     d_k+1 = -g_k+1 + beta_k*d_k
    d = -g + beta*d;
end
```

19.9 DFP Method

```
function x = dfp(Q,b)
%dfp - This function is used to minimize a positive definite
% quadratic form of n-variables using Devidon Fletcher
% Powell method.
% The objective function is
% Min 0.5*x'Qx - b'x, with Q positive definite
% Inputs:
%   Q, b : Coefficients in the objective function
% Output:
%   x    : Optimal solution.

n = length(b);
x = zeros(n,1);

g = Q*x - b;      % g_0
S = eye(n);       % S_0 = I
d = -S*g;         % d_0 = -S_0*g_0; g_k=grad(f(x_k));
k = 0;

while ((k < n) && (norm(g) ~= 0))
    %     Calculating h(alpha)=C(1)*alpha_k^2 + C(2)*alpha_k + C(3)
    C = [0.5*(d'*Q*d) (0.5*d'*Q*x + 0.5*x'*Q*d - b'*d) ...
        (0.5*x'*Q*x - b'*x)];
    %   alpha = golden_section(0,10000,0.001,C);
    alpha = - C(2)/(2*C(1));

    %     x_k+1 = x_k + alpha_k*d_k
    x = x + alpha*d;

    %     p_k = alpha_k*d_k=(x_k+1-x_k);
    p = alpha*d;

    %     q_k = g_k+1 - g_k
    q = (Q*x - b) - g;

    %     A_k = (p_k*p_k') / (p_k'*q_k)
    A = p*p' / (p'*q);

    %     B_k = - (S_k*q_k*q_k'*S_k) / (q_k'*S_k*q_k)
```

```
    B = - (S*q*q'*S) / (q'*S*q);

    %     S_k+1 = S_k + A_k + B_k
    S = S + A + B;

    %     Calculating g_k+1
    g = Q*x - b;  %g_k=grad(f(x_k))

    %     d_k+1 = - S_k+1 * g_k+1
    d = - S*g;

    k = k+1;
end
```

19.10 Frank and Wolfe's Method

```
%    Determines an optimal solution of linearly constrained NLP
%     via Frank and Wolfe's method
%     - The function frank_wolfe provides an optimal solution of
%       the problem.
%       Min  f(x)
%       s.t Ax=y, x>=0
%Inputs: (for f(x)=x^T Qx+b^Tx
%A, y, Q, b, x0 (initial point)
%Coefficients in the objective function are in the decreasing
%order for eg.
%To minimize x1^2+x1x2+x2^2+x1+x2
% subject to x_1+x_2+2x_3=3; x_1,x_2,x_3 >= 0 we have

%Q=[1 0.5 0.5;0.5 1 0;0.5 0 0.5];
%b=[-4;-3;-2];
%A=[1 1 2];
%y=[3];
%x0=[1;0;1];

%output: x : Optimal solution

function x0 = frank_wolfe(Q,b,A,y,x0)

k=0;
[m,n]=size(A);
type_obj=input(' 1 for min 0 for max =');
type_var=2;%type of variables
%typr_var is 2 for >=0 variables
%(<= or >= or unrestricted  (1/2/3)).
type_cons=3;%type of constraints
%type = vector of 1 or 2 or 3,i.e, <= or >= or =  (1/2/3)

M=[type_var*ones(m,1)];
N=[type_cons*ones(m,1)];
n = length(b);

x = zeros(n,1);

x=x0;
```

```
j=0;
x=[];
while (j~=1)
    c=2*Q*x0+b; %2*Q*x0+b is the gradient of the objective
    %function x^T Qx+b^T. For the general case, the gradient has
    %to be computed explicitly and that has to be taken as vector c.
    [opt_val,x,status]=SimplexTwoPhasemethod(c,A,y,M,N,type_obj);
    x=x';
    if (abs(c'*x - c'*x0)>0.001)
        fprintf('As c*x < c*x0, thus')
        for i=1:n
            fprintf('Value of variable %d  =  %6.2f \n',i,x0(i));
        end
        disp('is not optimal hence one more iteration is req.')
        eps=0.001;
        d=(x-x0);
        C=[d'*Q*d; (2*d'*Q*x0 + b'*d); (x0'*Q*x0 + b'*x0)];
        alpha = - C(2)/(2*C(1));
        if alpha<0 || alpha>1 % alpha has to lie b/w [0,1]
            h1=C(3);
            h2=sum(C);
            if h1<h2
                alpha=0;
            else
                alpha=1;
            end
        end
        % alpha = golden_section(0.01, 1, eps,C);
        x = x0 + alpha*d;
        x0=x;
        k = k+1;
    else
        disp('the optimal solution is:')
        for i=1:n
            fprintf('Value of variable %d  =  %6.2f \n',i,x0(i));
        end
        j=1;
    end
end
disp('The number of iterations used to solve this problem')
disp(k)
```

19.11 General Comments

1. The code for the golden section rule has been written for the polynomial case only. Also the codes for the steepest descent method, the conjugate gradient method, and Frank and Wolfe's method have been written for the quadratic case only. These codes can certainly be modified for the general nonlinear case (subject to appropriate convexity assumptions) provided the functions and their gradient/Hessian are input appropriately by employing symbolic functions of MATLAB. The interested readers are encouraged to use the "help" facility of MATLAB in this regard.
2. Purely from the user's point of view MATLAB has a very useful toolbox, namely Optimization Toolbox, which contains standard programs, e.g. linprog (for linear programming), quadprog (for quadratic programming) etc.
3. The readers are encouraged to write their own codes by combining some of the codes given here. For example, one can write the code for Gomory's cutting plane method by suitably combining the codes for the simplex method and the dual simplex method.
4. An appropriate source for learning MATLAB related to optimization is the text *Applied Optimization with MATLAB programming (P. Venkataraman, 2001, Wiley).*

References

1. F. Alizadeh, *Semidefinite programming homepage*, http://karush.rutgers.edu/~alizadeh/Sdppage/index.html
2. R. Almgren and N. Chriss, Optimal execution of portfolio transactions, *Journal of Risk*, Vol. 3, pp. 5-39, 2001.
3. M. Ammann, *Credit Risk Evaluation*, 2nd edition, Springer Verlag, 2001.
4. B. Aouni and O. Kettani, Goal programming model: a glorious history and a promising future, *European Journal of Operational Research*, Vol. 133, pp. 225-231, 2001.
5. H. S. H. Aretbi, Global routing for VLSI standard cells, *Electrical and Computer Engineering*, Vol. 1, pp. 485-488, 2004.
6. M. Avriel, *Nonlinear Programming*, McGraw Hill, 1969.
7. M. Avriel, W. E. Diewert, S. Schaible and I. Zang, *Generalized Concavity*, Plenum Press, 1988.
8. A. D. Bain, The choice of parameters in an NMR experiment, application to the inversion recovery T_1 method, *Journal of Magnetic Resonance*, Vol. 89, pp. 153-160, 1990.
9. R. Baldick, *Applied Optimization Formulation and Algorithms for Engineering Systems*, Cambridge University Press, 2006.
10. M. Bartholomew-Biggs, *Nonlinear Optimization with Financial Applications*, Kluwer Academic Publishers, 2003.
11. M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley, 1979.
12. M. S. Bazaraa, J.J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows*, John Wiley, 1990.
13. C. R. Bector, S. Chandra and J. Dutta, *Principles of Optimization Theory*, Narosa Publishing House, 2005.
14. R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
15. R. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, 1962.
16. A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, SIAM Series on Optimization, Philadelphia, 2001.

17. K. P. Bennett and O. L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, *Optimization Methods and Software*, Vol. 1, pp. 23-34, 1992.
18. C. Berge and A. Ghouila-Houri, *Programming Games and Networks*, John Wiley and Sons, 1965.
19. D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.
20. D. P. Bertsekas, *Nonlinear Programming, 2nd edition*, Athena Scientific, 1999.
21. D. Bertsimas and A. W. Lo, Optimal control of execution costs, *Journal of Financial Markets*, Vol. 1, pp. 1-50, 1998.
22. J. F. Bonnans, J. C. Gilbert, C. Lemarechal, and C. A. Sagastizabal, *Numerical Optimization, Theortical and Practical Aspects*, Springer Verlag, 2000.
23. J. C. G. Boot, *Quadratic Programmming: Algorithms-Anomalies Applications*, North-Holland Publishing Company, 1964.
24. K. H. Borgwardt, *The Simplex Method: A Probabilistic Analysis, Algorithms and Combinatorics*, Springer Verlag, 1987.
25. J. M. Borwein, Proper efficient points for maximizations with repect to cones, *SIAM Journal of Control and Optimization*, Vol. 15, pp. 57-63, 1977.
26. J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer Verlag, 2000.
27. S. Boyd and L. Vandenberge, *Convex Optimization*, Cambridge University Press, 2004.
28. D. Brigo and F. Mercurio, *Interest Rate Models: Theory and Practice*, Springer Verlag, 2001.
29. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, Vol. 2, pp. 1-43, 1998.
30. R. E. Burkard, A general Hungarian method for the algebraic transportation problem, *Discrete Mathematics*, Vol. 22, pp. 219-232, 1978.
31. M. Capinski and T. Zastawniak, *Mathematics of Finance: An Introduction to Financial Engineering*, Springer Verlag, 2003.
32. C. W. Caroll, The created response surface technique for optimizating nonlinear restrained systems, *Operations Reserach*, Vol. 9, pp. 169-184, 1961.
33. V. Chankong and Y. Y. Haimes, *Multiobjective Decision Theory and Methodology*, Elsevier Science, 1983.
34. A. Charnes and W. Cooper, *Management Models and Industrial Applications of Linear Programming*, John Wiley, 1961.
35. A. Charnes and W. W. Cooper, Programming with linear fractional functions, *Naval Research Logistics Quaterly*, Vol. 9, pp. 181-186, 1962.
36. D. S. Chen and S. Zionts, An exposition of the group theoritic approach to integer linear programming, *Opsearch*, Vol. 9, pp. 75-102, 1972.

37. C. C. N. Chu and D. F. Wong, A quadratic programming approach to simultaneous buffer insertion/sizing and wire sizing , *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, pp. 787-798, 1999.
38. M. Clerc, *Particle Swarm Optimization*, ISTE Publishing Company, 2006.
39. G. Cornuejols and R. T üt ünc ü, *Optimization Methods in Finance*, Cambridge University Press, 2007.
40. B. D. Craven, *Fractional Programming*, Heldermann-Verlag, 1988.
41. C. C. Chang and C. J. Lin, *LIBSVM : a library for support vector machines*, http://www.csie.ntu.edu.tw/ ~ cj lin/libsvm/
42. N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel based Learning Methods*, Cambridge University Press, 2000.
43. Y. H. Dai and R. Fletcher, New algorithms for singly linear constrained quadratic programming subject to lower and upper bounds, *Mathematical Programming* (ser. A), Vol. 105, pp. 403-421, 2005.
44. G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, 1963.
45. L. David, *Credit Risk Modeling*, New Age International, 2007.
46. K. M. Deb, *Optimization for Engineering Design: Algorithms and Examples*, Prentice Hall of India, 2003.
47. J. -L. Deneubourg, S. Aron, S. Goss and J. -M. Pasteels, The self-organizing exploratory pattern of the Argentine ant, *Journal of Insects Behavior*, Vol. 3, pp. 159-168, 1990.
48. W. Dinkelbach, On nonlinear fractional programming, *Management Science*, Vol. 13, pp. 492-498, 1967.
49. L. C. W. Dixon, *Nonlinear Optimization*, The Englisa Universities Press, 1972.
50. L. C. W. Dixon, *Optimization in Action*, Academic Press, 1976.
51. M. Dorigo and T. St ützle, *Ant Colony Optimization*, MIT Press, 2004.
52. R. R. Egudo, Efficiency and generalized convex duality for multiobjective programs, *Journal of Mathematical Analysis and Applications*, Vol. 138, pp. 84-94, 1989.
53. E. J. Elton, M. J. Gruber, S. J. Brown and W. Goetzmann, *Modern Portfolio Theory and Investment Analysis*, 6th edition, John Wiley, 2002.
54. S. C. Fang and S. Puthenpura, *Linear Optimzation and Extension: Theory and Algorithms*, Prentice Hall, 1993.
55. W. Fenchel, *Convex Cones, Sets and Functions*, Lecture Notes, Princeton University Press, 1951.
56. J. A. Ferland, Quasiconvexity and pseudoconvexity of functions on the nonnegative orthant, In Generalized Concavity in Optimization and Economics (eds. S. Schaible and W. T. Ziemba), Academic Press, pp. 169-182, 1981.
57. A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, 1968.

58. R. Fletcher, *Practical Methods of Optimization, Unconstrained Optimization*, John Wiley and Sons, 1980.
59. A. M. Frieze, Algebraic linear programming, *Mathematics of Operations Research*, Vol. 7, pp. 172-182, 1982.
60. G. Fung and O. L. Mangasarian, Proximal support vector machines, Proceedings KDD-2001, San Francisco, pp. 77-86, 2001.
61. G. Fung, M. Dundar, J. Bi and B. Rao, A fast iterative algorithm for Fisher discriminant using heterogeneous kernels, Proceedings of the 21st International Conference on Machine Learning ACM, Canada, pp. 264-272, 2004.
62. G. Fung, O. L. Mangasarian and Shavlik J, Knowledge-based support vector machine classifiers, Technical Report 01-09, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Wisconsin, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-09.ps, 2002.
63. D. Gale, *The Thoery of Linear Economic Models*, McGraw-Hill, 1960.
64. R. S. Garfinkel and M. R. Rao, The bottelneck transportation problem,*Naval Research Logistic Quarterly*, Vol. 18, pp. 465-472, 1971.
65. R. S. Garfinkel, and G. L. Nemhauser, *Integer Programming*, Wiley, New York, 1972.
66. S. L. Gass, *Linear Programming: Methods and Applications*, McGraw-Hill, 1975.
67. A. M. Geoffrion, Proper efficiency and the theory of vector maximization, *Journal of Mathematical Analysis and Applications*, Vol. 22, pp. 618-630, 1968.
68. P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, Academic Press, 1981.
69. F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.
70. H. J. Greenberg and W. P. Pierskalla, A review of quasi-convex functions, *Operations Research*, Vol. 19, pp. 1553-1570, 1977.
71. S. R. Gunn, Support vector machines for classification and regression, Technical Report, School of Electronics and Computer Science, University of Southampton, Southampton, 1998. On-line at http://www.isis.ecs.soton.ac.uk/resources/svminfo/
72. G. Hadley, *Linear Programming*, Addison-Wesley, 1962.
73. K. B. Haley, The multi-index problem, *Operations Research Quaterly*, Vol. 11, pp. 368-369, 1963.
74. P.L. Hammer, Time minimizing transportion problems, *Naval Research Logistic Quarterly*, Vol. 16, pp. 345-357, 1969.
75. M. A. Hanson, On sufficiency of the Kuhn-Tucker conditions, *Journal of Mathematical Analysis and Applications*, Vol. 80, pp. 545-550, 1981.
76. F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, Tata McGraw Hill Publishing Co., 2001.
77. J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
78. T. C. Hu, *Integer Programming and Network Flows*, Addison-Wesley Publishing Company, 1975.

79. A. Husain and K. Gangiah, *Optimization Techniques for chemical Engineers*, Macmillan Company of India Ltd., 1976.
80. M. D. Intriligator, *Mathematical Optimization and Economic Theory*, Prentice Hall, 1971.
81. J. P. Ignizio, *Linear Programming in Single & Multiple Objective Systems*, Prentice Hall, 1982.
82. J. Jahn, *Vector Optimization: Theory, Applications and Extensions*, Springer Verlag, 2004.
83. Jayadeva, R. Khemchandani and S. Chandra, Twin support vector machines for pattern classification, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 29, pp. 901-911, 2007.
84. V. Jeyakumar, A note on strong duality in convex semidefinite optimization: necessary and sufficient conditions, *Optimization Letters*, Vol. 2, pp. 15-25, 2008.
85. V. Jeyakumar and B. Mond, On generalized convex mathematical programming, *Journal of Australian Mathematical Society (Series B)*, Vol. 34, pp. 43-53, 1992.
86. T. Joachims, Making large-scale SVM learning practical, In Advances in Kernel Methods Support Vector Learning (eds. B. Sch öolkopf, C. Burges and A. Smola), MIT Press, pp. 169184, 1999.
87. T. Joachims, *SVMLIGHT: support vector machine*, http://www-ai.cs.uni-dortmund.de/SOFTWARE/SVM_LIGHT/svm_light.html
88. P. Kall, *Stochastic Linear Programming*, Springer Verlag, 1976.
89. N. S. Kambo, *Mathematical Programming Techniques*, East-West Press, 1991.
90. S. Karlin, *Mathematical Methods and Theory of Games, Programming and Economics*, Addison-Wesley Publishing Company, 1959.
91. N. Karmarkar, A new polynomial time algorithm for linear programming, *Combinatorica*, Vol. 4, pp. 373-395, 1984.
92. V. Kecman, T. -M. Huang and M. Vogt, Iterative single data algorithm for training kernel machines from huge data sets: theory and performance, In Support Vector Machines: Theory and Applications, (eds. L. Wang), Springer Verlag, 2005.
93. S. Keerthi, S. Shevade, C. Bhattacharyya and K. Murthy, Improvements to SMO algorithm for SVM regression, Technical Report CD-99-16, Department of Mechanical and Production Engineering, National University of Singapore, 1999. http://citeseer.ist.psu.edu/shevade99improvements.html
94. R. Khemchandani, *Mathematical Programming Applications in Machine Learning*, PhD thesis, Department of Mathematics, Indian institute of Technology Delhi, 2008.
95. R. Khemchandani, Jayadeva and Suresh Chandra, Knowledge based proximal support vector machines, *European Journal of Operational Research*, 2007. DOI: 10.1016/j.ejor.2007.11.023.
96. R. Khemchandani, Jayadeva and Suresh Chandra, Optimal kernel selection of twin support vector machines, *Optimization Letters*, Vol. 3, pp. 77-88, 2009.

97. R. Kissell and M. Glantz, *Optimal Trading Strategies: Quantitative Approaches for Managing Market Impact and Trading Risk*, AMACOM, American Management Association, 2003.
98. W. Kuo, V. R. Prasad, F. A. Tillman and C. L. Hwang, *Optimal Reliability Design, Fundamentals and Applications*, Cambridge University Press, 2001.
99. P. J. van Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, 1992.
100. G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui and M. I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research*, Vol. 5, pp. 27-72, 2004.
101. L. S. Lasdon, *Optimization Theory for Large Systems*, Macmillan, 1970.
102. L. S. Lasdon, J. Plummer, B. Buehler and A. D. Waren, Optimal design of efficient acoustic antenna arrays, *Mathematical Programming*, Vol. 39, pp. 131-155, 1987.
103. X. J. Lin, A tutorial on cross-layer optimization in wireless networks, *IEEE Journal on Selected Areas in Communications*, Vol. 24, pp. 1452-1463, 2006.
104. C. F. Lin and S. D. Wang, Fuzzy support vector machines with automatic membership setting, In Support Vector Machines: Theory and Applications, (eds. L. Wang), Springer Verlag, 2005.
105. D. T. Luc, *Theory of Vector Optimization*, Springer Verlag, 1989.
106. D. G. Luenberger, *Optimization by Vector Space Methods*, John Wiley, 1969.
107. D. G. Luenberger, *Investment Science*, Oxford University Press, 1998.
108. T. L. Magnanti, Fenchel and Lagrange duality are equivalent, *Mathematical Programming*, Vol. 7, pp. 253-258, 1974.
109. O. L. Mangasarian, *Nonlinear Programming*, McGraw Hill Book Co., 1969.
110. O. L. Mangasarian, Generalized support vector machines, In Advances in Large Margin Classifiers, (eds. A. J. Smola, P. Bartlett, B. Schökopf and D. Schuurmans), MIT Press, pp. 135-146, 2000.
111. O. L. Mangasarian and D. R. Musicant, Active support vector machine classification, Technical Report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Wisconsin, 2000.
112. O. L. Mangasarian and E. W. Wild, Multisurface proximal support vector classification via generalized eigenvalues, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, pp. 69-74, 2006.
113. H. M. Markowitz, Portfolio selection, *Journal of Finance*, Vol. 7, pp. 77-91, 1952.
114. B. Martos, *Nonlinear Programming* , North-Holland, 1975.
115. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, Equations of state calculations by fast computing machines, *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092, 1953.
116. M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1996.
117. K.G. Murty, *Linear Programming*, John Wiley and Sons, 1983.

118. J. von Neumann and O. Morgenstern, *Theory of Games and Economc Behaviour*, Princeton University Press, 1944.
119. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Verlag, 1999.
120. G. Owen, *Game Theory*, Academic Press, 1995.
121. C. Van De Panne, *Methods for Linear and quadratic Programming*, North-Holland Publishing Co., 1975.
122. D. T. Phillipse, A. Ravindram and J. Solberg, *Operations Research: Principles and Practice*, Wiley, 1976.
123. A. T. Phillips and J. B. Rosen, A quadratic assignment formulation of the molecular conformation problem, *Journal of Global Optimization*, Vol. 4, pp. 229-241, 1994.
124. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, 1982.
125. C. Papadimitriou, *Computational Complexity*, 1st edition, Addison Wesley, 1999.
126. S. R. Paranjape, The simplex method: two basic variables replacement, *Management Sciences*, Vol. 12, pp. 135-141, 1965.
127. P. M. Pardalos and M. G. C. Resende, *Handbook of Applied Optimization*, edited volume, Oxford University Press, 2002.
128. T. Parthasarathy and T. E. S. Raghavan, *Some Topics in Two Person Games*, Elsevier Publishing Company, 1971.
129. J. Ponstein, Seven kinds of convexity, *SIAM Review*, Vol. 9, pp. 115-119, 1967.
130. M. V. Ramana, An exact duality theory for semidefinite programming and its complexity implications, *Mathematical Programming*, Vol. 77, pp. 129-162, 1997.
131. A. W. Roberts and D. E. Varberg, *Convex Functions*, Academic Press, 1973.
132. S. Roman, *Introduction to the Mathematics of Finance: From Risk Management to Options Pricing*, Springer Verlag, 2004.
133. S. Ross, *An Introduction to Mathematical Finance: Options and Other Topics*, Cambridge University Press, 1999.
134. P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs, *Journal of Computer and Systems Sciences*, Vol. 37, pp. 130-143, 1988.
135. R. L. Rardin, *Optimization in Operations Research*, Pearson Education Pvt. Ltd., 1998.
136. R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.
137. P. A. Samuelson, *Foundations of Economic Analysis*, Harvard University Press, 1955.
138. H. Salkin, *Integer Programming*, Addison-Wesley Publishing Company, 1975.
139. P. S. Sastry, An introduction to support vector machines, In Computing and Information Science: Recent Trends, (eds. J. C. Mishra), Narosa Publishing House, 2003.
140. Y. Sawaragi, H. Nakayama and T. Tanino, *Theory of Multiobjective Optimization*, Academic Press, 1985.

141. S. Schaible and T. Ibaraki, Fractional programming: invited review, *European Journal of Operational Research*, Vol. 12, pp. 325-338, 1983.
142. P. S. Shelokar, V. K. Jayaraman and B. D. Kulkarni, Ant algorithm for single and multiobjective reliability optimization problems, *Quality and Reliability Engineering Optimization*, Vol. 18, pp. 497-514, 2002.
143. D. M. Simmons, *Nonlinear Programming for Operatinal Research*, Prentice-Hall, 1975.
144. C. Singh, Optimality conditions in multiobjective differentiable programming, *Journal of Optimization Theory and Applications*, Vol. 53, pp. 115-123, 1987.
145. I. N. Sneddon, *Fourier Transforms*, Courier Dover Publications, 1995.
146. M. S. Sodhi, LP modelling for asset liability management: a survey of choices and simplifications, *Operations Research*, Vol. 53, pp. 181-196, 2005.
147. I. M. Stancu-Minasian, A sixth bibliography of fractional programming, *Optimization*, Vol. 55, pp. 405-428, 2006.
148. R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*, John Wiley, New York, 1986.
149. R. E. Steuer, Y. Qi and M. Hirschberger, Portfolio optimization: new capabilities and future methods, *Zeitschrift fr Betriebswirtschaft*, Vol. 76, pp. 199-219, 2006.
150. J. Stoer and C. Witzgall, *Convexity Optimization in Finite Dimensions I*, Springer Verlag, 1970.
151. J. A. K. Suykens and J. Vandewalle, Least squares support vector machines classifiers, *Neural Processing Letters*, Vol. 9, pp. 293-300, 1999.
152. W. Swarc, Some remarks on the time transportation problem, *Naval Research Logistic Quarterly*, Vol. 18, pp. 475-485, 1971.
153. K. Swarup, Linear fractional functional programming, *Operations Research*, Vol. 12, pp. 1029-1036, 1965.
154. H. A. Taha, *Operations Research: An Introduction*, Prentice Hall, 1997.
155. H. A. Taha, *Integer Programming: Theory Applications and Computations*, Academic Press, 1975.
156. J. S. Taylor and N. Cristianini, *Kernel Methods for Pattern Anlaysis*, Cambidge University Press, 2004.
157. L. C. Thomas, Mathematical programming and its applications in finance, In Mathematical Programming and Game Theory for Decision Making, (eds. S. K. Neogy, R. B. Bapat, A. K. Das and T. Parthasarthy), World Scientific, pp. 1-14, 2007.
158. L. C. Thomas, D. B. Edelman and J. N. Crook, *Credit Scoring and its Applications*, SIAM, 2002.
159. M. J. Todd, Semidefinite optimization, *Acta Numerica*, Vol. 10, pp. 515-560, 2001.
160. L. Vandenberghe and S. Boyd, Semidefinite Programming, *SIAM Review*, Vol. 38, pp. 49-95, 1995.
161. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.
162. V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.

163. T. Weir and B. Mond, Generalized concavity and duality, Generalized Concavity in Optimization and Economics (eds. S. Schaible and W. T. Ziemba), Academic Press, New York, pp. 263-280, 1981.
164. W. B. White, S. M. Johnson and G. B. Dantzig, Chemical equilibrium in complex mixtures, *Journal of Chemical Physics*, Vol. 28, pp. 751-755, 1958.
165. W. L. Winston *Operations Research: Applications and Algorithms*, Wadsworth Publishing Company, 1997.
166. D.A. Wismer, *Optimization Methods for Large Scale Systems*, McGraw-Hill, 1971.
167. P. Wolfe, A duality theorem for nonlinear programming, *Applied Mathematics*, Vol. 19, pp. 239-244, 1961.
168. H. Wolkowicz, R. Saigal and L. Vandenberghe (eds.), *Handbook of Semidefinite Programming*, Kluwer Academic Publishers, 2000.
169. Q. Yang, A new proof of strong duality theorem of semidefinite programming, *Journal of Mathematical Analysis and Applications*, Vol. 303, pp. 622-626, 2005.
170. X. S. Yang, *Nature Inspired Metaheuristic Algorithms*, Luniver Press, 2008.
171. L. Y. Yu, X. D. Ji and S. Y. Wang, Stochastic programming models in financial optimization: a survey, *Advance Modelling and Optimization*, Vol. 5, pp. 1-26, 2003.
172. S. Zionts, *Linear and Integer Programming*, Prentic-Hall, 1974.
173. W. I. Zangwill, *Nonlinear Programming: a Unified Approach*, Prentice Hall, 1969.

Index

adjacent corner point, 23
adjacent extreme points, 63
affine scaling algorithms, 436
all integer linear program, 218
alternative optima, 45
ant colony optimization, 547
artificial variable, 34
aspiration level, 495
asset, 579
augmented Lagrangian, 404
average case probabilistic computational complexity, 436

balanced transportation problem, 153
Balas' additive algorithm, 244
barrier function, 388
barrier function method, 363
basic constraint qualification, 306
basic equation of optimality, 378
basic feasible solution, 19
basic solution, 19
basic variable, 19
basis matrix, 19
Bellman's equation, 397
Bellman's optimality principle, 396
beta of asset, 602
beta of portfolio, 602
big-M method, 37
Boltzmann probability, 540
bond angle, 619
bordered Hessian matrix, 446
branch and bound method, 236

capital asset pricing model, 596
capital market line, 598
centering transformation, 421
Charne's and Cooper algorithm, 462
chemical equilibrium, 618
chromosome, 542
closed half spaces, 62
coding scheme, 543
complementary slackness conditions, 113
complementary slackness theorem, 111
computational complexity, 411, 536
cone, 512
cone of positive semi-definite matrices, 514
conic programming problem, 514
conjugate directions, 344
conjugate gradient method, 348
constraint qualification, 280
convergence rate, 327
converse duality theorem, 109
convex combination, 63
convex cone, 513
convex function, 255
convex hull, 62
convex optimization problem, 267
convex set, 61
crossover, 544
curse of dimensionality, 402
curve fitting, 621
cycling, 52

degenerate basic feasible solution, 19
descent function, 394
descent property, 326

deviational variables, 497
Dinkelbach's algorithm, 471
direct duality theorem, 109
divisibility, 582
doubly stochastic matrix, 192
dual cone, 523
dual simplex method, 120
dynamic programming, 395

E-optimality, 627
edge, 63
efficient frontier, 484
efficient solution, 484
eigen value problem, 522
energy function, 620
epigraph, 259
error minimizing problem, 556
exact penalty function method, 404
existence theorem, 111
expected pay-off, 134
extreme point, 62

Farkas lemma, 309
fathomed, 238
feasible direction, 300
feasible set, 12
feasible solution, 12
Fibonacci method, 333
Fisher information matrix, 626
fitness function, 543
Fletcher and Reeves method, 352
forward path, 549
fractional programming problem, 461
Frobenius inner product, 515
fundamental theorem of matrix games, 135

genetic algorithm, 542
Geoffrion properly efficient solution, 487
global min point, 300
globally convergent, 326
goal, 496
goal deviation, 496
goal function, 497
golden section ratio, 330
Gomory's cut constraint, 222
Gomory's cutting plane method, 221, 230

hard margin classifier, 563
Hungarian method, 197
hyperplane, 62
hypograph, 259

ideal penalty function, 375
ideal solution, 480
improperly efficient solution, 487
infeasible, 14
integer linear programming problem, 217
interior point method, 436

Karmarkar's projective scaling algorithm, 426
kernel function, 573
KKT optimality conditions, 308, 460
Klee and Minty cube, 415

Lagrange Function, 308
Lagrange multipliers, 308
level set, 259, 441
lexicographic minimum vector, 499
linear fractional programming problem, 461
linear goal programming problem, 499
linear independence constraint qualification, 312
linear matrix inequality, 516
linearizing cone, 302
liquidity, 582
local min point, 299
logarithmic barrier function, 407
Lorentz cone, 514

machine learning, 553
margin, 561
market portfolio, 598

Markowitz bullet, 594
Markowitz curve, 591
Markowitz efficient frontier, 592
Markowitz theory, 579
maxmin principle, 134
Mercer's theorem, 574
metaheuristic, 538
method of column minima, 166
method of Lagrange multipliers, 276
midpoint convex function, 294
minimum risk weight line, 592
mixed integer linear program, 218
multiobjective programming problem, 479
multistage decision problems, 395
mutation, 545

n-simplex, 63
network routing problem, 395
NMR experiment, 625
non-basic variable, 19
nondegenerate basic feasible solution, 19
nonlinear fractional programming problem, 469
NP problem, 537

offspring, 542
one fund theorem, 598
order of convergence, 327

P problem, 537
Pareto solution, 484
pattern classification problem, 554
penalty function method, 363
permutation matrix, 192
pheromone, 547
pointed cone, 513
polyhedron, 63
polynomial time algorithm, 413, 537
polytope, 63
population, 542
portfolio, 582
post optimality analysis, 125
primal-dual pair, 103
projection matrix, 370
projective transformation, 420
properly efficient solution, 486
protein folding problem, 619
proximal support vector machine, 576
pseudoconcave function, 454
pseudoconvex function, 454
pseudolinear function, 457
pure strategies, 134

quadratic termination property, 326
quasi-convex functions, 262
quasiconcave function, 442
quasiconvex function, 442
quasilinear function, 450

reliability, 622
return, 580
return path, 550
revised simplex method, 82
risk, 580
risk-free asset, 580
risky asset, 580
routing problem, 612

saddle point, 134
scalarization, 494
Schur complement, 518
security market line, 602
self dual cone, 524
semi-definite programming problem, 516
sequential goal programming technique, 501
sequential quadratic programming, 405
sequential unconstrained minimization technique, 363
shadow prices, 119
short selling, 581
simplex algorithm for fractional programs, 462
simplex method, 26

simulated annealing, 538
slack variables, 16
soft margin classifier, 568
solution of the game, 135
stigmergy, 547
strategy space, 133
strict local max point, 274
strict local min point, 274
strictly concave function, 257
strictly convex function, 257
strictly feasible point, 421
strictly feasible SDPP, 530
strictly pseudoconcave function, 458
strictly pseudoconvex function, 458
strictly quasiconcave function, 452
strictly quasiconvex function, 452
strong duality for SDPP, 528
strong duality in semi-definite programming, 531
sufficient optimality conditions, 460
support vector, 565
support vector machine, 565
supporting hyperplane, 62
surplus variables, 16
symbolic matrix, 157
symmetric duality, 103
synthesis antennae array problem, 616

test data, 562
trace function, 515
tradeoff, 480
training data, 562
transhipment problem, 209
travelling salesperson problem, 210
twin support vector machine, 577
two fund theorem, 595
two person zero-sum matrix game, 133
two phase method, 34

unbalanced transportation problem, 183
unbounded solution, 14
unimodal, 327
unimodular matrix, 156
usable direction, 301
usable feasible direction, 301
utility function, 444

value of the game, 134
VLSI design, 609

weak duality for semi-definite programming, 528
weak duality theorem, 108
weak efficient solution, 482
weighted sum approach, 489
wire-sizing problem, 614
wireless network problem, 624
Wolfe dual, 316
Wolfe's method, 285

Numerical Optimization with Applications

Suresh Chandra
Jayadeva
Aparna Mehra

Numerical Optimization with Applications provides a focused and detailed study of various numerical optimization methods and their applications in Science, Engineering and Management. Apart from discussing standard optimization methods and their traditional applications, the book includes some very recent topics like Semi-definite Programming, Second Order Cone Programming, Evolutionary Methods and Global optimization. An attempt has been made to present some modern and non-conventional applications of numerical optimization in the areas of Machine Learning, VLSI Design/ Electrical Circuits and Financial Mathematics. A distinctive feature of the book is also to provide basic MATLAB codes as building blocks for readers to develop their own codes for various algorithms discussed in the book.

ISBN 978-81-7319-854-0
9 788173 198540

Narosa
Publishing House
www.narosa.com